Delay/Encoder Specifications

##

This is the specification of the Delay/Encoder module to be used on the silicon strip detector readout system for E771 and E789. Comments and questions can be delivered to Hector L. Gonzalez at Ext 2773.

##

Table of Contents

1. General Information

1.1. Purpose

The front end readout electronics for the Silicon Strip Detector is designed to process data at the RF bucket frequency, 53MHz. The Delay/Encoder(DE) module has been specified to accept data at 53MHz, provide a delay mechanism while a trigger decision is made, and generate an address hit list upon a Level 1 accept signal. A simplified block diagram is provided in Figure 2.

The delay element continuously stores data while the level 1 system is processing data corresponding to previously stored events. The delay is implemented in RAM and it is required for the control system to map the level 1 decisions into an eight bit address. The current implementation assumes that processing of an event takes about 1μsecond from the time that it is loaded into the DE. It is mandatory that the event acceptance be time ordered and the decision time be fixed with respect to the event occurrence. The address of accepted events is broadcasted by the Master Timing Controller to all Sequencers modules in the system and each Sequencer addresses the DE in its crate. The addressing mechanism triggers the DE to read the event from memory and transfer it to the encoder section.

The data encoding scheme uses the trigger bucket and the previous bucket simultaneously to generate an address hit list. A flag is asserted whenever the previous bucket has the bit set for the address been output. The address hit list is transmitted synchronously to a crate Sequencer module which serves as a crate controller and event builder for two planes of silicon strip data. The Sequencer is capable of transmitting hit data over fiber optic at 40Mbytes/sec or being readout through Fastbus.

This document includes figures and timing diagrams intended to simplify the specifications. In some cases, specifically the timing diagrams, the information is an attempt to specify the module and its interface with other system components.

1.2. Silicon Strip Readout System

This section presents a simplified block diagram of the silicon strip readout system. Figure 1 shows the interconnection of all the

modules that are referenced in this document. A brief description of each module follows.

PC - PostAmp/Comparator board, 12 per crate. Processes 128 pre-amp silicon strip signals, outputs discriminated data to the Delay/Encoder and outputs analog and digital sums to Level 1.

DE - Delay/Encoder board, 12 per crate. Provides event buffering for the PC discriminated data and for level 1 accepted events transmits a hit list to the Sequencer.

SEQ - Sequencer board, 1 per crate. Fans out system clock to PC and DE, initiates the encoding of a event, stores, pipelines and transmits encoded events to the next level. The events can be readout through FASTBUS.

FSCC- Fastbus Smart Crate Controller board, 1 per crate. Initializes the crate by excercising control over the SEQ, runs local diagnostics and provides an alternate data path to readout events.

MTC - Master Timing Control board, 1 per silicon strip readout system. The MTC synchronizes the 12 SEQ on the system by providing timing and control. Some of the functions that it performs are listed below:

- Distributes the RF clock to all SEQs.
- Maps a level 1 accept signal into an address of the DE memory and transmits addresses to all SEQs.
- Queue level 1 accepted events.
- Controls the write enable signal for the DE.
- Responds to READY and ERROR condition from the SEQs.
- Interface with the overall experiment controller.
- Synchronize System.

The readout system consists of 12 readout crates and a control crate that contain the MTC and other special modules. Each of the readout crates processes two planes of silicon strip data. Data processing is done in groups of 128 strips by a PC and DE pair. The 12 DEs in a crate send data, in parallel, to the crate SEQ. For a formal description of the system refer to the 'Silicon Strip Readout Implementation Plan' document.

In the context of this specification an **event** is the output of the PC and they are generated every 18.9nanoseconds.

1.3. Application

The DE is being designed for the Silicon Strip Readout System for E771 and E789. The function of the module is hardwired and there is no other application for it beyond the ones described on this document.

1.4. Packaging

The board is a single width FASTBUS module that does not implement any FASTBUS protocol.

1.5. Power Requirements

The worse case estimated current for the module are listed below:

Voltage	Current	Power
-5.2V	17A	86W
-2.0V	5A	10W
+5.0V	<1A	

The estimated -5.2 volts typical current is around 15 amps.

1.6. Cooling Requirements

The module will operate at the temperature range provided by the FASTBUS cooling system.

2. Theory of Operation and Operating Modes

The DE module is a single width board packaged in FASTBUS that does not implement a FASTBUS interface. The DE accepts PC discriminated data from 128 silicon strips, provides event buffering, encodes and transmits accepted events to a crate SEQ. The module communicates with the PC and SEQ through a special FASTBUS auxiliary backplane. The auxiliary connector signals for the DE are described in Appendix A.

The module is divided in two independent functions; the Delay and the Encoder, see Figure 2. The following sections provide a brief description for each function.

2.1. Delay

The Delay element receives 128 channels of discriminated data from the associated PC and provides buffering for upto 256 events (~4.8 μseconds). During data acquisition the DE continuously stores data in a FIFO like memory, while the level 1 system is making decisions for previously stored events. The DE does not implement any logic to prevent overwriting interesting events. This operation is delegated to the MTC which keeps track of the system write pointer (for DE) and the events been queued. For system implementation reasons, it is mandatory that event acceptance be time ordered and the decision time be fixed with respect to the event occurrence.

The delay element control logic requires that the 53MHz input (CLK2) be a 50% duty cycle clock. The logic splits the 18.9 nanoseconds time slice of each bucket into a read and write periods for a combined bandwidth of 106MHz. The write operation uses an address counter clocked by CLK2 and a write enable signal (WRITE*, asserted low) generated by the MTC and distributed in each crate by the SEQ.

The WRITE* signal is send 128 cycles before the SYNC* pulse. The DEs retime this signal using the CLK2 and SYNC* signals for synchronizing the start of event acquisition, see timing 2. The DEs will track their synchronization by checking that the write address is zero when the SYNC* signal is asserted. Note that prior to the assertion of WRITE* the DEs had been reset, which forces the write address to zero.

The readout of an event occurs when the MTC receives an accept pulse from the level 1 trigger system. The MTC maps the

pulse into an address for the DEs memory and sends the address to all SEQs in the system. The address generated by the MTC shall correspond to the previous bucket location. The previous and accepted buckets are loaded into registers and the Encoder is enabled to begin encoding that particular event.

2.2. Encoder

The Encoder is a simultaneous two bucket hit-list address generator. The encoding is performed in two stages, byte and bit levels, see Figure 3. At load time, the byte encoder performs byte integration, see Figure 4. The output of this process is a 16-bit word with a bit set for bytes with hit channels. At encoding time the byte encoder sequentially selects bytes to be processed by the bit encoder. The bit encoder (block ENCODER-8) loads the input data when ready and outputs a byte wide address stream of asserted bits. The encoded address is formed by concatenating the byte address with the bit address. In addition the bit encoder sets a flag (DATA0) whenever the previous bucket has a hit for the address being output. The hit list is generated from low to high address and no hit count is generated by the DE. An example of byte encoding is shown below:

Bit #	Previous Bucket	Trigger Bucket	Hit Type	Bit ADD (HEX)	Flag
7	1	0	A	7	1
6	0	0	-	-	-
5	0	1	B	5	0
4	1	1	C	4	1
3	0	0	-	-	-
2	0	0	-	-	-
1	1	1	C	1	1
0	0	1	B	0	0

Note: The encoder never looks at addresses without hits, this is represented with "-".

The alternative to remove type A hits is implemented on the prototype.

There are two options for the address hit list data transfer, 53MHz or 26.5MHz, CLK2 and CLK3 respectively. These options are switch selectable for each DE. Its implementation requires that the skew on the rising edge of CLK2 and CLK3 be kept to less than 2 nanoseconds.

3. Input/Output Specifications

The DE is a two port module, PC port and a SEQ port, See Figure 2. Both of these ports are implemented on an application specific Fastbus Auxiliary backplane that pairs a PC with a DE and has separate connections between each DE and the SEQ.

3.1. Communication Interfaces

3.1.1. PostAmp/Comparator Port

The PC port is a 128 bit uni-directional single-ended ECL connection from a PC to the associated DE. Synchronization of this port is controlled by the SEQ supplied CLK1 and CLK2 clocks. CLK1 is remotely programmed through FASTBUS and the rising edge is used by the PC to latch the silicon strip discriminated data. CLK2 is referenced (delayed) to CLK1 such that the PC output data is valid while CLK2 is high at the DE, see Timing 1. It is required that data on this port be valid for at least 12 nsececonds simultaneously at all DE in a crate.

3.1.2. Sequencer Port

The Sequencer port is the access port for the SEQ to readout the level 1 accepted events. This data port is a byte wide point-to-point connection designed to support a 53MHz data transfer rate.

The port provides an address bus, a data bus and control signals. To initiate a transfer the Sequencer supplies the DE with an event address and asserts the Add_Valid signal. After a fixed delay the Encoder will assert a Data_Valid signal (if hits present) and start transmitting the address hit list to the SEQ until completion. The data transfer is synchronous with CLK2 or CLK3 depending on the

user selected encoder operating frequency. A non-detailed timing diagram is provided in Timing 4.

3.1.3. Front Panel

The DE front panel is intended to provide information that will help diagnose problems on the DE. The prototype provides test points for the previous and trigger bytes been processed by the bit-encoder and the byte mask. The byte mask provides a bit for each of the 16 bytes of the 128 channels. A bit is asserted for the bytes that have a hit. As the bytes are processed the bits of the mask are cleared.

4. Initialization

Initialization of the DEs is achieved by asserting the RESET signal. Then the MTC initiates the enable of the write process and after the appropriate delay the enable of the Level 1 system.

After reset, the write counter points to location zero, the Encoder is in the ready state and all control signals driven by the DE are negated. The read counter is not initialized because it is loaded on demand when an event is accepted.

5. System and Module Diagnostics

5.1. Hardware Test

The DE does not implement any internal diagnostic tests or FASTBUS interface to access it's memory. The decision of excluding these features is based on timing and power considerations.

For debugging, a test module that emulates the PC output port and the SEQ was designed. The tester will be operated by an intelligent FASTBUS master, i.e. the FSCC or TSC.

5.2. System Test

For system diagnostics the PC should be able to supply data patterns that the DE stores into it's memory and events to be encoded are requested through the SEQ.

6. Appendix A

This appendix describes the DE auxiliary connector signals used on the silicon strip readout crates. In addition any other signals of interest are described.

CLK1 - A 53MHz clock driven by the SEQ and used by the PC to latch data. This clock is remotely programmable through the FASTBUS port on the SEQ.

CLK2 - A 53MHz clock driven by the SEQ. The clock is a delayed version of CLK1 used to synchronize the write process in the DE with the output data of the PC, to generate internal timing and as a reference when transmitting data to the SEQ at 53MHz.

CLK3 - A 26.5MHz clock driven by the SEQ. An alternate clock used to transmit data to the SEQ.

DI(0:127) - Input discriminated data driven by the associated PC. A 100 ohms termination is provided by the DE.

Address(0:7) - The address bus (bussed to all DE) driven by the SEQ to transfer event addresses to the DE.

Add_Valid - Signal driven by SEQ to validate ADD(0:7).

Write_En* - Write enable signal distributed on the backplane by the SEQ. The signal is controlled by the MTC or through a FASTBUS register on the SEQ.

Sync* - Synchronization signal generated by the MTC and used by the DEs to test write counter synchronization at each zero crossing. If a DE has a write counter different from zero then it is out of synchronization.

Sync_Err	-	Signal asserted by a DE that is out of synchronization and received by the SEQ. The signal is wire-ored on the backplane.
Reset	-	Reset signal distributed on the backplane by the SEQ.
Data_X(0:7)	-	Encoded hit list data bus for Delay/Encoder X, where X is a hex number assigned to each DE. Terminated by the SEQ.
Data_ValidX	-	Data valid signal asserted by Delay/Encoder X, where X is a hex number assigned to each DE. Terminated by the SEQ.

appendix B

This appendix is intended to explain the system diagnostics involving the PC output port, the DE and the SEQ. There are two system test of interest. The first one involves the complete silicon strip readout system (starting at PC), the other one performs test on selected crates. Details about these test have not been worked out and the implementation steps presented here should be modified as development of the involved modules progress. A series of steps follows:

Step1: The FSCC enable/disable specific PCs and select the data pattern type, i.e. static or dynamic. Also, it initializes the SEQ for global or local test.

Step2: The FSCC reset the DE through the SEQ.

Step3: Simultaneously enable the PC to generate patterns and the DE to start recording events. This is, the PC receives a signal of similar timing to the DE write enable signal.

Step4: Write an event address to the SEQ if the test is local or to the MTC if the test is global.

Step5: For local test, the FSCC reads the event from the SEQ. For global test the SEQ sends the event to the next level.

Step6: Check the event and display/record errors.

Step7: If test enable, go to Step1 or Step 4.

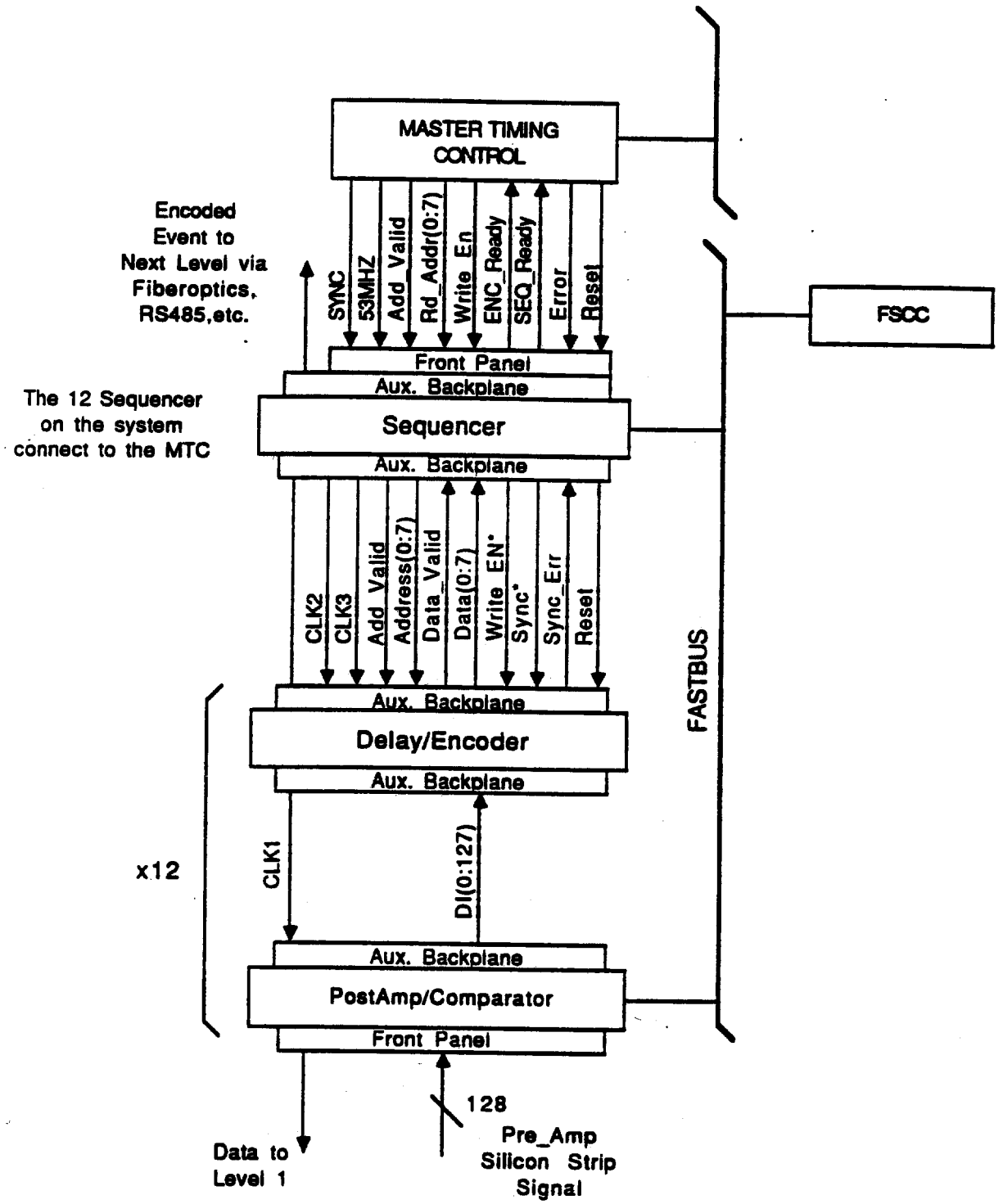
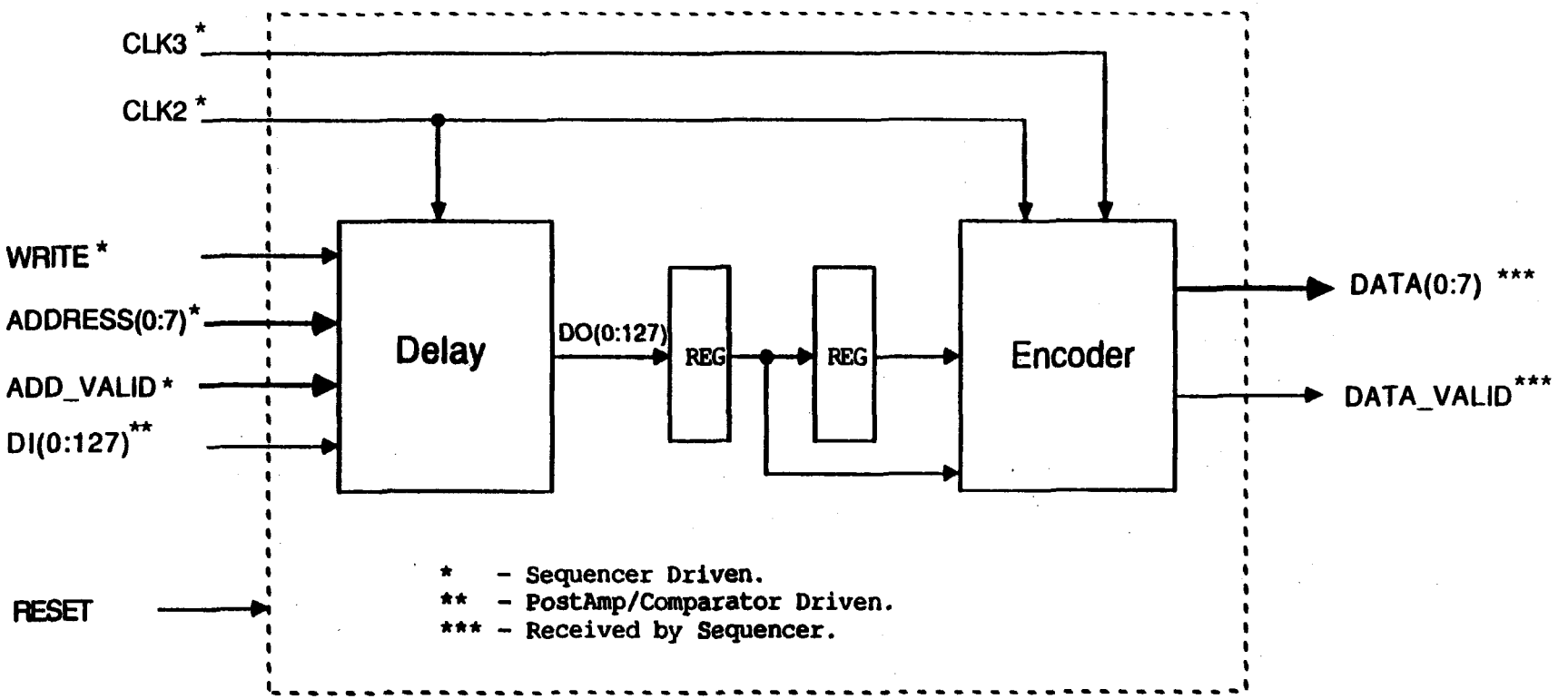


FIGURE 1: Silicon Strip Readout System



PC PORT:

DI(0:127)	- PC output data.

SEQ PORT:

WRITE	- Write enable for the delay memory.
ADDRESS(0:7)	- Accepted bucket address.
ADD_VALID	- Validates ADDRESS(0:7).
DATA(0:7)	- Channel to transmit the address hit list.
DATA_VALID	- Enable the Sequencer to clock data into its FIFO.

Figure 2: Delay/Encoder Block Diagram

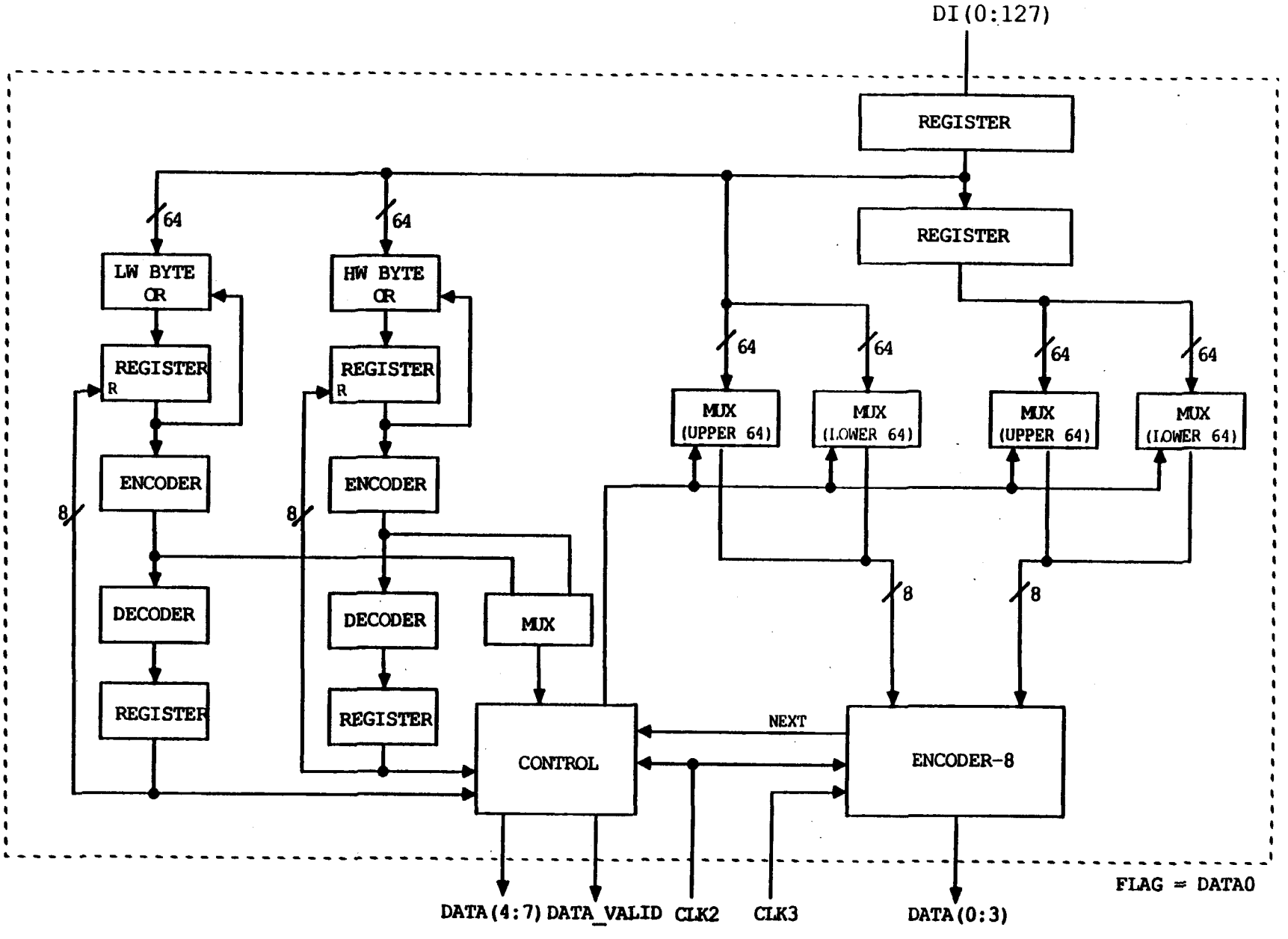


Figure 3: 128-BIT ENCODER

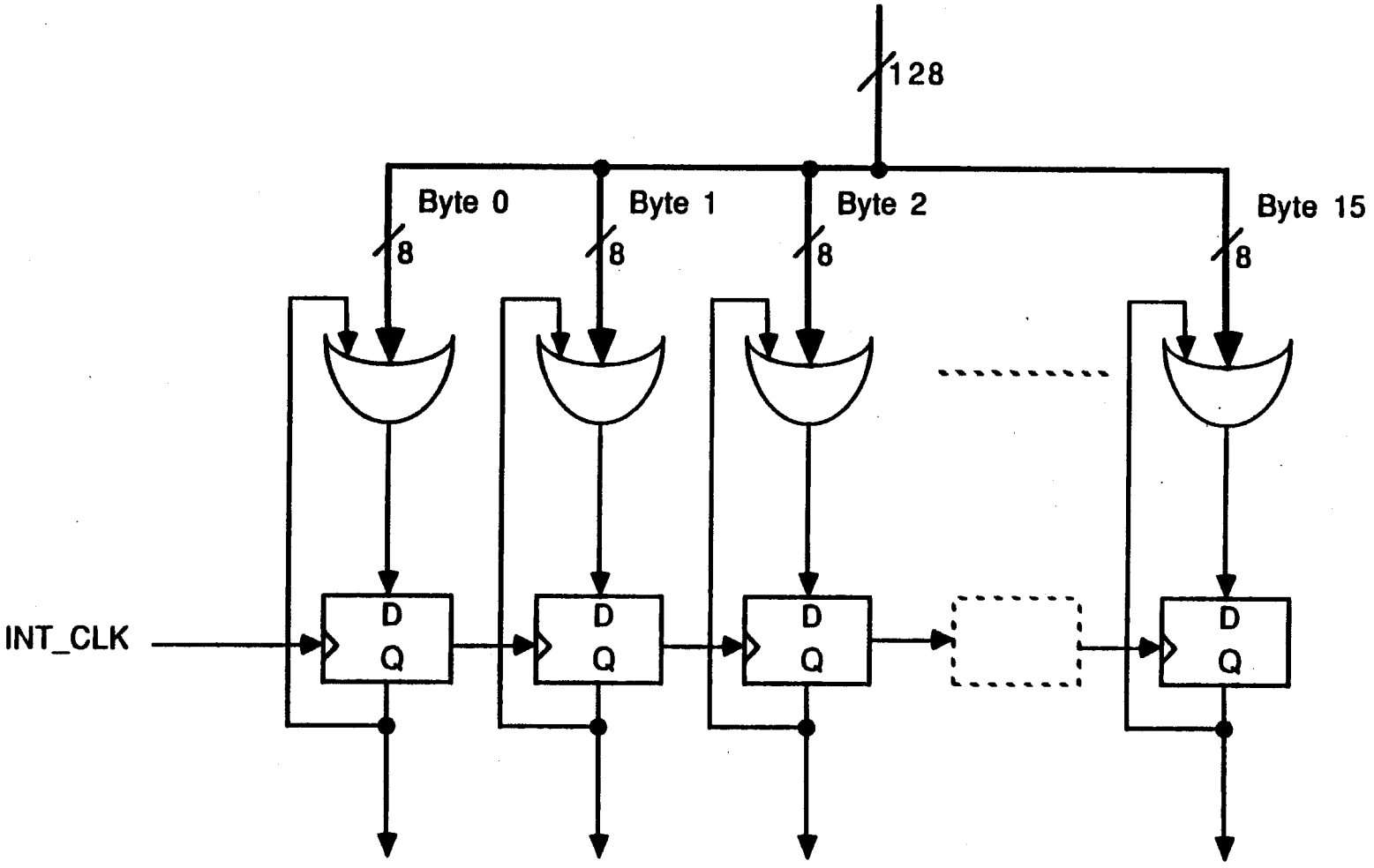
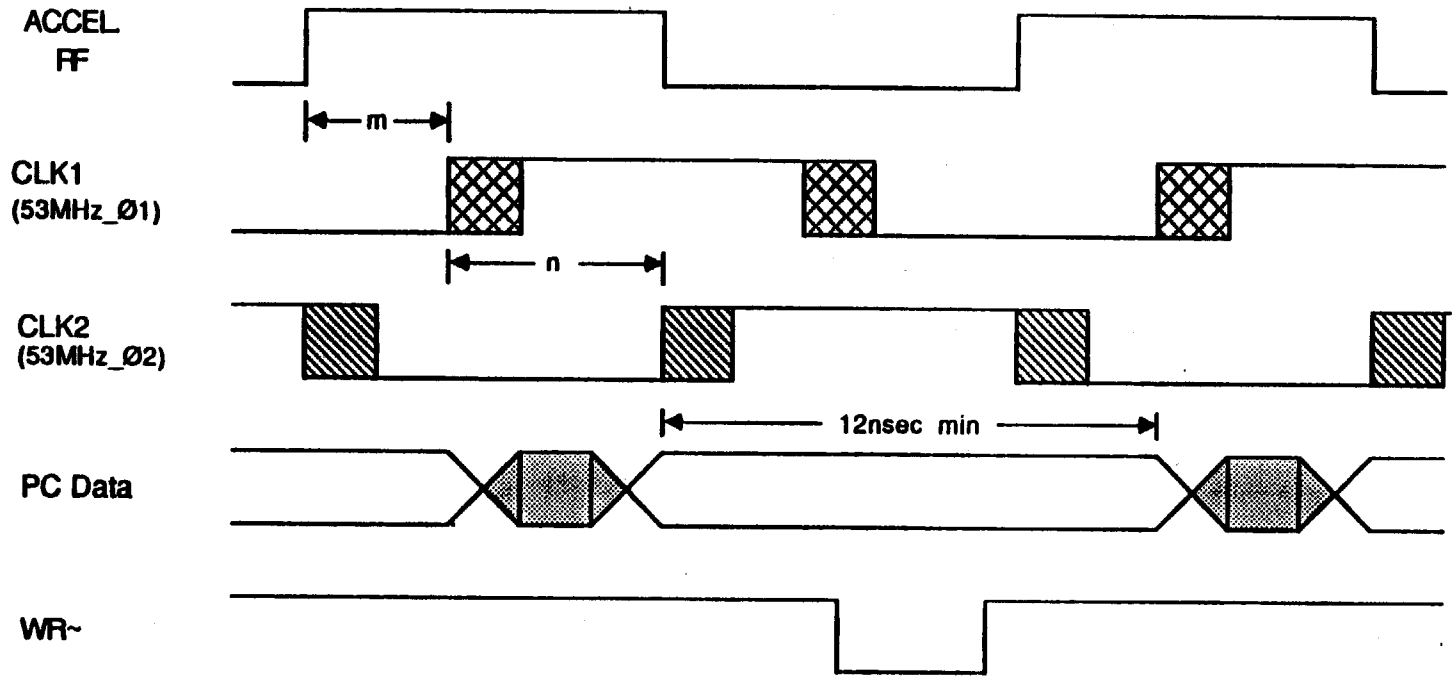
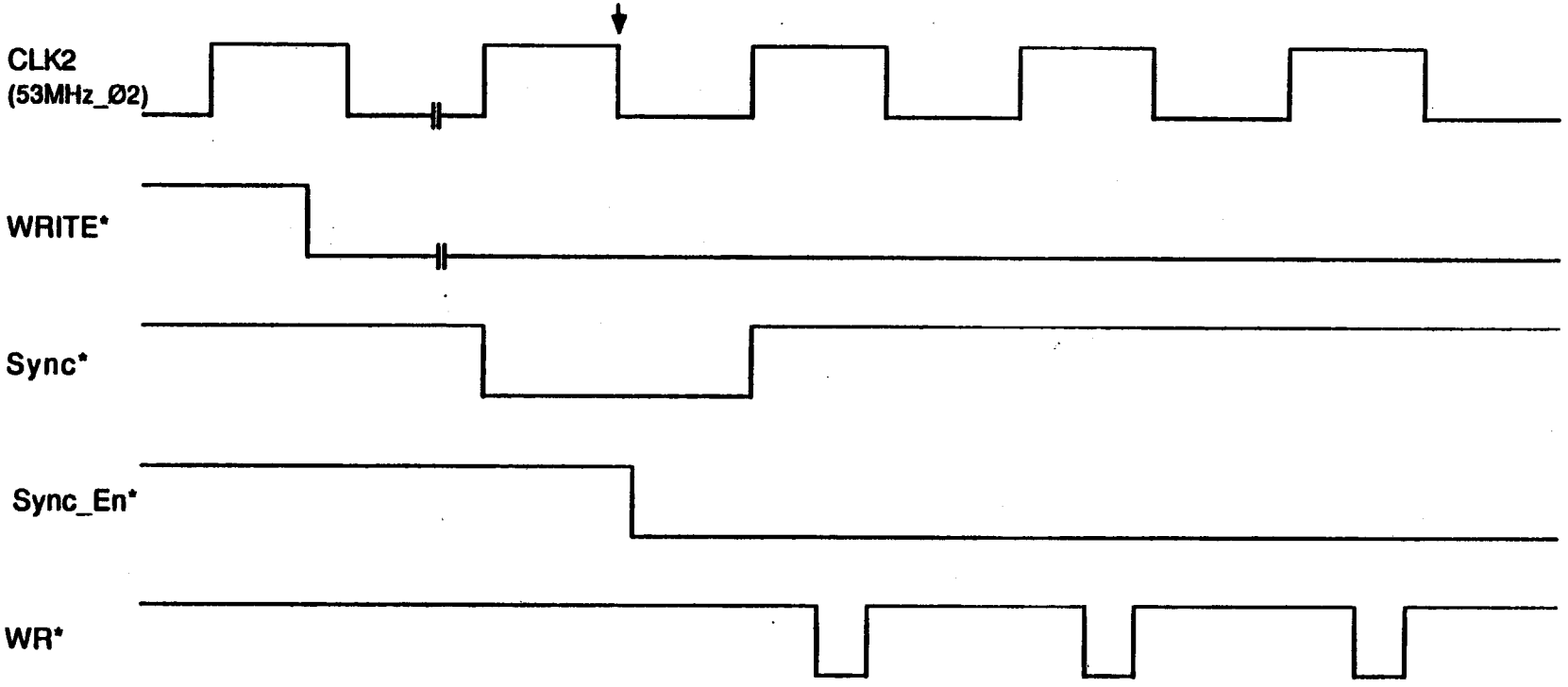


Figure 4: Byte Integration



m - Delay added to synchronize the PC to the accelerator RF.
n - Delay added to CLK1 clock to synchronize the DE to the PC output data.

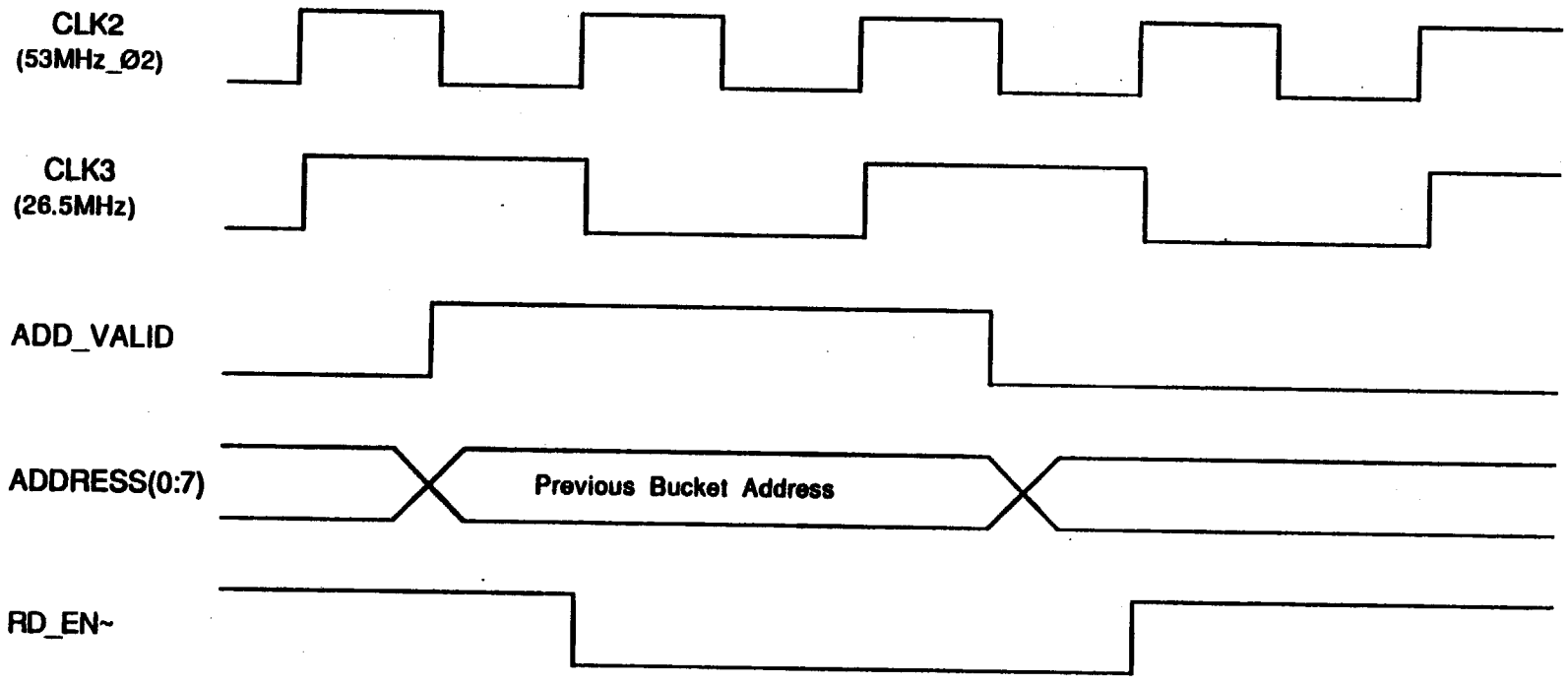
Timing 1: Synchronization of DE to PC.



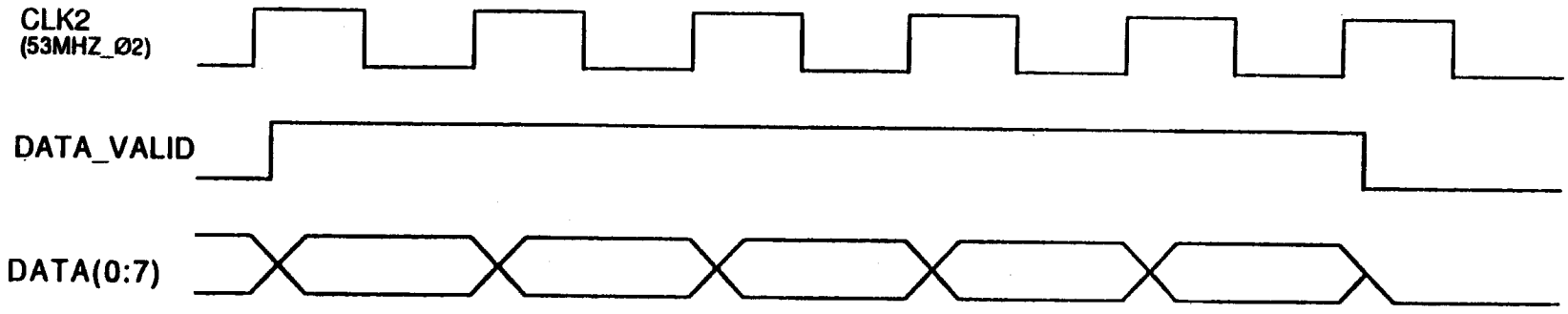
DESCRIPTION:

WRITE* - Write enable signal on the auxiliary backplane driven by Sequencer.
SYN_EN* - DE internally synchronized write enable.
WR~ - Write pulse to memory.

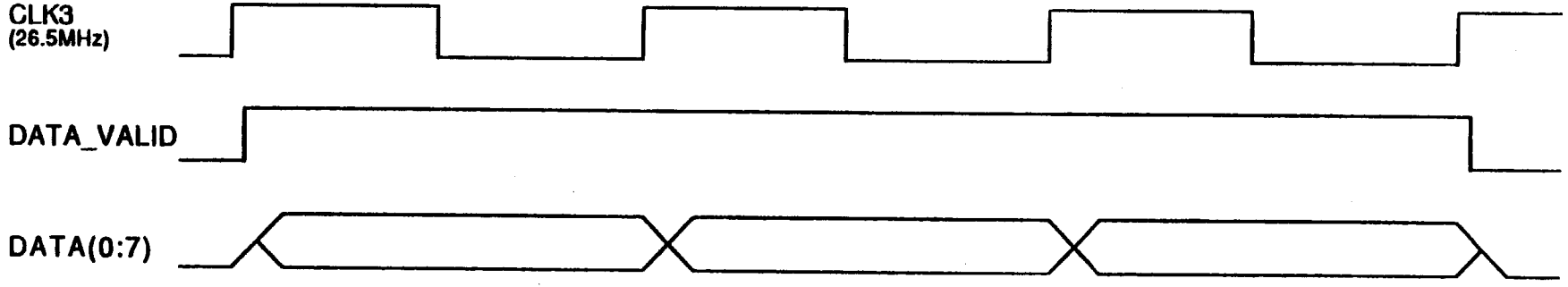
Timing 2: Write Enable Synchronization.



Timing 3: Address Transfer from SEQ to DE.



Timing 4a : 53MHz Encoded Event Transfer



Timing 4b: 26.5MHz Encoded Event Transfer

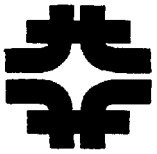
Fermi National Accelerator Laboratory

January 3, 1991

TO: Distribution
FROM: Carl Swoboda
SUBJECT: Sequencer Hardware Description

The attached document is the "as built" hardware description for the SSD Sequencer module. PLease add this document to your SSD Readout System binder.

Distribution:
David Christian
Brad Cox (E771)
Peter Garbincius
Franco Grancagnolo (E771, MS 219)
Hector Gonzalez
Bill Haynes
Rick Kwarciany
Wolfgang Kowald (E771)
Mark Larwill
Cash McManus (E771)
Garry Moore
Tom Nash
John Peoples
Ruth Pordes
Matteo Recagni (E771)
Robert Trendler
Ken Treptow



Fermi National Accelerator Laboratory

Silicon Strip Detector Readout System

SEQUENCER MODULE HARDWARE DESCRIPTION

R. DeMaat, M. Larwill, A. Romero

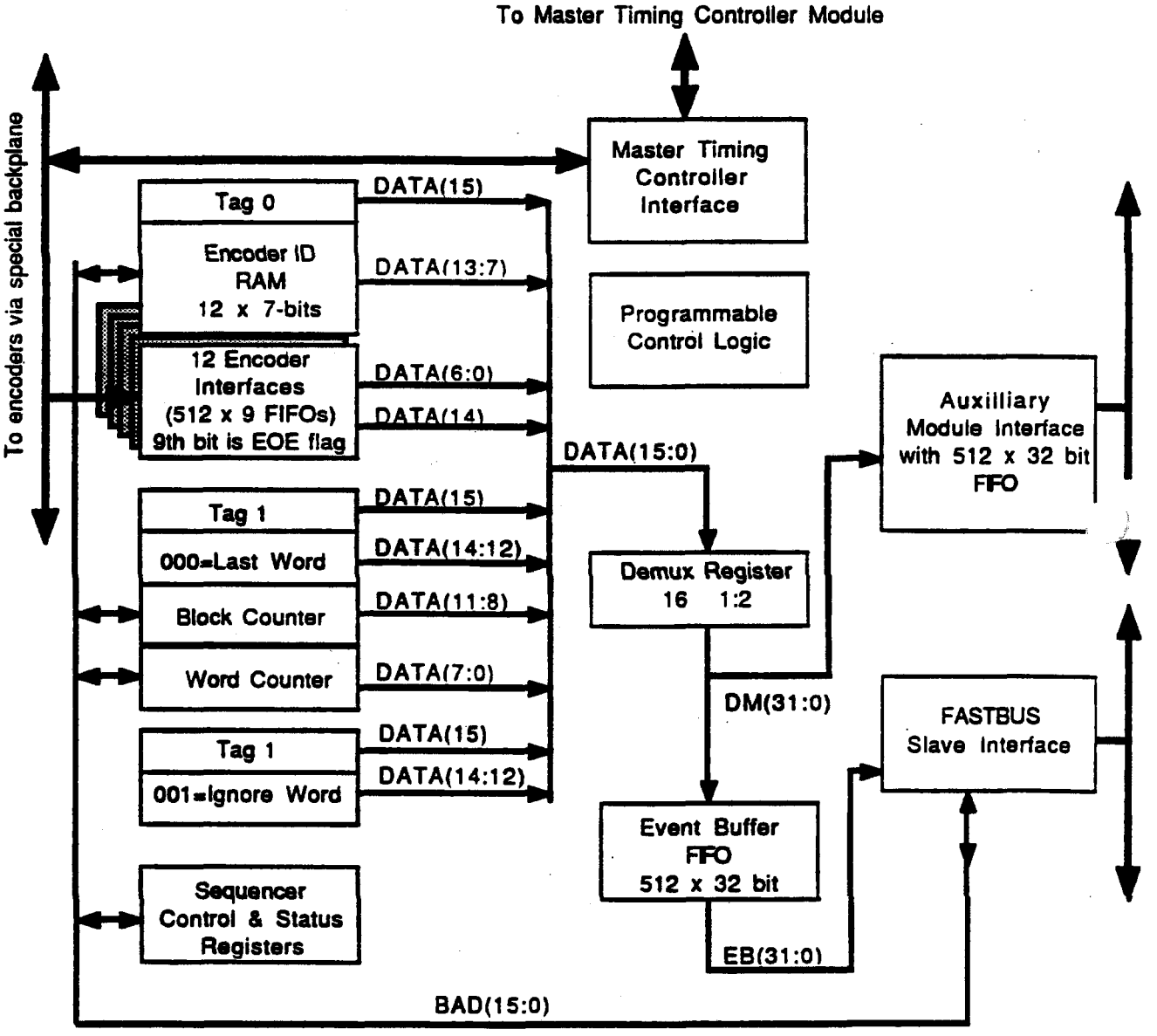
December 6, 1990

Fermilab Document Number HN 99

1. GENERAL INFORMATION

1.1. Purpose

This document describes the "Silicon Strip Detector Readout System Sequencer Module" hereafter referred to as the sequencer. As one component of a larger system, this module accepts silicon strip detector data from the twelve encoder modules in its FASTBUS crate, combines the data and outputs it to other system components via FASTBUS or via an auxiliary link such as RS-485, ECLine or fiber optics. The general block diagram of the sequencer is as follows:

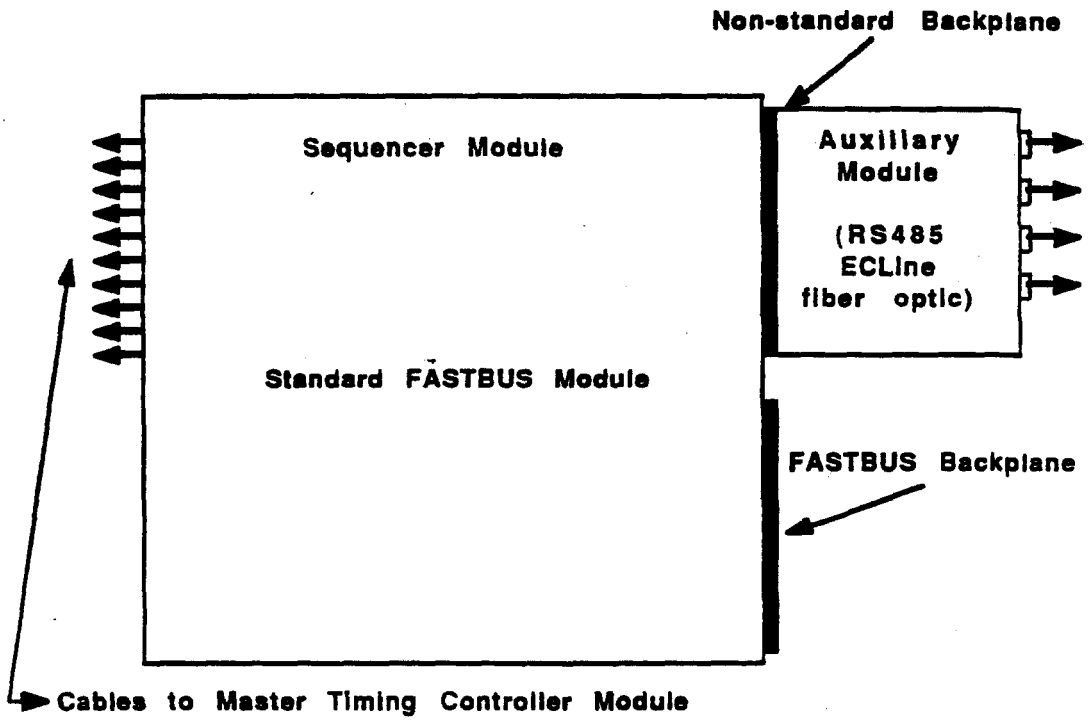


1.1.1. Standard Bus System Used

The sequencer is a single width FASTBUS module and includes FASTBUS slave capability. The FASTBUS slave is used for the initialization of some programmable parameters on the module and may be used to read out the encoded silicon strip detector data.

In addition to the standard FASTBUS interface, the sequencer uses a non-standard backplane on the FASTBUS auxiliary connector position to communicate with the encoder and postamp/discriminator modules. This backplane is optimized for high speed parallel transfers.

Some of these FASTBUS auxiliary connector pins are not used by the special backplane to communicate with the other modules. These pins are used instead by the sequencer to communicate with an I/O link driver such as RS-485, ECLine or fiber-optics. A special auxiliary interface module plugs into the back of the FASTBUS crate for this function. The figure describes this pictorially.

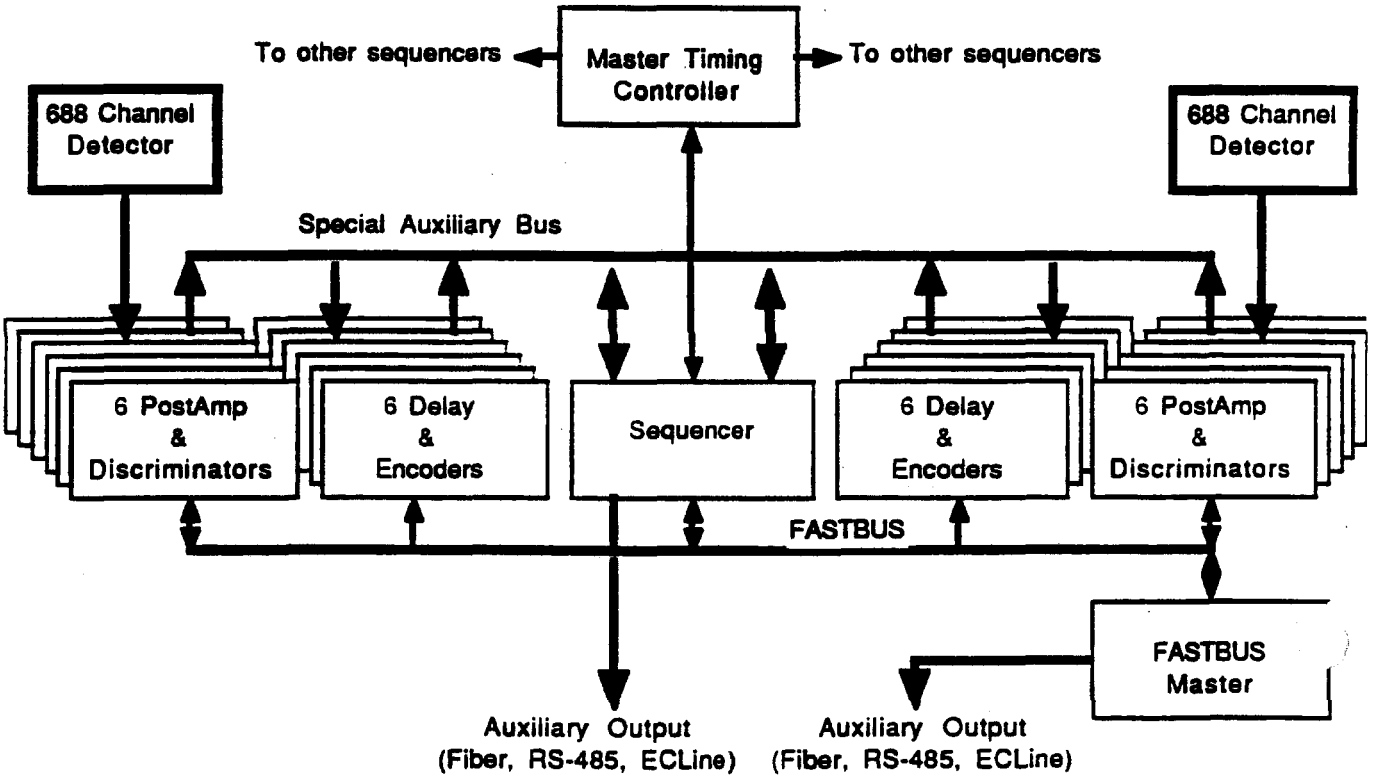


1.1.2. Number of Channels

Data comes into the sequencer from 12 encoder modules in the FASTBUS crate. The data is concatenated and may leave the sequencer via FASTBUS or auxiliary link (RS-485, ECLine or fiber optics).

1.2. Application

The sequencer module sits in the middle of the FASTBUS crate and communicates with the FASTBUS encoder modules via the special auxiliary backplane. Front panel connectors on the module connect via cables to the master controller to provide system clock, trigger commands, system RESET, errors and etc. The sequencer sends its data out via a rear mounted auxiliary card. Data may also be read via FASTBUS transfers. The sequencer may be used at Fermilab in experiment E771 and may be adaptable to other applications. In the E771 application, 24 planes of detector will be used requireing 12 crates of readout electronics. Each crate of electronics will be configured as pictured here:



1.3. Packaging

1.3.1. Module

The sequencer is implemented on a standard single width FASTBUS module which is 15.878" X 14.437" plus front panel. The module plugs into a FASTBUS crate which is then installed in an equipment rack along with the FASTBUS power and cooling system.

If the auxiliary port is used to output data, then a FASTBUS standard size auxiliary module is also used which contains logic and interface hardware to provide a high speed RS-485, ECLine or fiber optic data link. This auxiliary module plugs into the rear of the backplane at the sequencer module position in the slot provided for it by the FASTBUS crate.

1.3.2. Front and Rear Controls Connectors and Displays

1.3.2.1. CLK2 Delay Switch

The 53 Mhz clock received via a front panel connector is delayed by a programmable delay line and sent to the post amp/discriminator modules as a signal named CLK1. CLK1 is delayed by another programmable delay line and sent to the encoder modules as a signal named CLK2. The total amount of delay of CLK1 and CLK2 is also introduced on a signal called SYNC. The programmable delay line for CLK1 is programmed by FASTBUS, but the delay line for CLK2 is set by a 6-bit DIP switch. This switch provides a binary progressive delay from 0 to 31.5 nsec in .5 nsec increments.

SLV ACC	The Slave Access LED illuminates whenever the Sequencer is accessed by FASTBUS
53 MHZ	53 MHZ clock input (NIM)
SYNC	SYNC signal input (NIM)
MTC I/O	34 pin MTC interface port
PRM OUT	Permit Out output conn.
PRM IN	Permit In input conn.
SYNCERR	Sync Error, LED
EV SIZE OVRFLOW	Event Size Overflow, LED
ENCFIFO OVRFLOW	Encoder Fifo Overflow, LED
ENC BSY	Encoder Busy, LED
EVINAUX	Event in Auxiliary Buffer Fifo, LED
EVINFB	Event in FASTBUS Buffer Fifo, LED
SEQ BSY	Sequencer Busy, LED
+5V	+5V present, LED
-5.2V	-5.2V present, LED
SEQ INTRNAL RESET	Sequencer internal circuitry reset button

1.3.2.2. Front Panel Displays

The sequencer module displays the following indicators of module operation and status on its front panel.

- FASTBUS SLAVE ACCESS; this LED indicator lights for a minimum of 100 msec each time a FASTBUS access to the sequencer occurs.

- SYNC ERROR, This LED indicator lights and remains lighted until RESET by FASTBUS any time an encoder module in the crate signals the sync error condition. Sync error is signalled by an encoder module if its write counter is not at the zero count when the sync signal is received from the Master Timing Controller via the sequencer.

- EVENT SIZE OVERFLOW, this LED indicator is lighted by the occurance of an event which produces more that 255 'hits'. The sequencer does not 'know' this error has occurred until the data has already been received into its encoder FIFOs and is being counted as it is read out to FASTBUS and/or the auxiliary port. The occurrance of this error does not justify having to initialize the system yet a problem exists in how to dispose gracefully of the extra data. In normal operation the control logic will truncate an event at 255 'hits', properly insert the 'LAST WORD' into the data stream, and remove all extra data from the encoder FIFOs without FASTBUS intervention. Optionally, under the control of a bit in CSR0, the control logic will output the additional data while allowing the word counter to wrap around. This will be useful for system tests.

- ENCODER FIFO OVERFLOW, This LED indicator lights and remains lighted until RESET by FASTBUS any time the encoder interface FIFOs are overfilled by data from the encoders. This is a system failure caused by the Master Timing Controller not using the SEQ_READY signal properly. When SEQ_READY is not asserted, the encoder data FIFOs are at least half full and the sequencer should not be given any new trigger addresses. At this time, there is room in the FIFOs to hold the data generated by a single trigger which may have been on its way to the sequencer before the Master Timing Controller received the SEQ_READY being deasserted. Any additional triggers sent by the Master Timing Controller after SEQ_READY is deasserted may generate enough data to overfill the FIFOs and generate the error.

- ENCODERS BUSY, This LED indicator is lighted by the logical 'OR' of the DATA VALID signals from each encoder in the crate. The signal is not 'stretched' or latched and normally occurs too fast to be seen. If this LED remains lighted, an encoder in the crate has failed and has effectively halted the system.

- EVENT IN AUXILIARY INTERFACE BUFFER, This LED indicator is lighted by the presence of data in the Auxiliary Interface FIFO. It will stay lit as long as that FIFO is not empty.

- EVENT IN FASTBUS BUFFER, This LED indicator is lighted by the presence of data in the FASTBUS event FIFO. It will stay lit as long as that FIFO is not empty.

- SEQUENCER BUSY, This LED indicator is lighted by the logical 'OR' of the FIFO half-full signals from each encoder FIFO ion the sequencer module. The signal is not 'stretched' or latched and normally occurs too fast to be seen. If this LED remains lit, the sequencer is not processing the FIFO data out to FASTBUS and/or the auxiliary interface and has effectively halted the system. This may be caused by data not being removed from the FASTBUS event buffer or by the auxiliary link not being ready, or by PERMIT IN not being asserted.

- +5 VOLT STATUS, this LED indicates that the board's +5 volt bus is powered up.

- -5.2 VOLT STATUS, this LED indicates that the board's -5.2 volt bus is powered up.

1.3.2.3. Front Panel Connectors

Signals marked '+' are 'party lined' in a single flat cable. One end connects to the Master Timing Controller (MTC) which terminates the signals. The sequencers attach at various places along the cable. The last sequencer module on the cable provides signal terminations.

- 53 Mhz CLOCK input, terminated NIM, from MTC.

+ WRITE ENABLE input, differential ECL, asynchronous to clock, from MTC. This signal is bussed to each encoder module over the auxiliary backplane. Enables encoders to accept hit data.

- SYNC input, terminated NIM, synchronous to clock, from MTC. This signal gets delayed by the same amount as CLK1 plus the same amount as CLK2 before being bussed to the encoder modules over the auxiliary backplane to test synchronization at each zero count.

+ RESET input, differential ECL, asynchronous to clock, resets control logic and fifos, bussed over the auxiliary backplane to the encoders where it resets write counters.

+ EVENT ADDRESS input, differential ECL, 8-bits, asynchronous to clock, bussed to encoder modules over the auxiliary backplane to designate a stored event. Note that this is actually the address of the event previous to the one requested by the trigger system. This is because the encoders are required to evaluate the previous RF bucket for 'hits' when encoding an event.

+ ADDRESS VALID input, differential ECL, asynchronous to clock, bussed to encoder modules over the auxiliary backplane to initiate encoding of a stored event.

+ SEQ_READY output, wire OR'd ECL, active low, signals that the sequencer has room in its fifos for events and thus can accept "read addresses".

+ ENC_READY output, wire OR'd ECL, active low, this is the sum of all encoder data valid signals. It signals to the master controller that the encoders are ready to accept another trigger.

+ ERROR output, wire OR'd ECL, active high, signals that the sequencer fifos have been overfilled by the encoders or that an encoder has lost synchronization. A fatal error requiring system RESET.

- PERMIT IN - Single ended TTL. A signal from the upstream module indicating that the sequencer may transmit. It's used to allow multiple sequencers to feed a party line.

- PERMIT OUT - Single ended TTL. A signal to a downstream module used to allow the next sequencer in a daisy chain to use the auxiliary party line if applicable.

1.3.2.4. Special Auxiliary Backplane Signals

- CLK1, delayed from the front panel 53 Mhz clock. Individually routed to each post amp/discrimminator module. The delay between the 53 Mhz clock received via the front panel, and this signal, is programmable via FASTBUS from 0 to 32 nsec in .5 nsec increments.

- CLK2, delayed from CLK1, individually routed to each encoder module. The delay between CLK1 and this signal is programmable via a 6-bit DIP switch from 0 to 32 nsec in .5 nsec increments.

- CLK3, 26.5 Mhz, CLK2 divided by two, bussed to all encoder modules for use as a data clock.

- RESET, Asynchronous, Bussed to all discriminators and encoder modules.

- WRITE ENABLE, Synchronized to 53 Mhz CLK2, bussed to all encoder modules, active low.

- EVENT ADDRESS, Asynchronous , 8-bits. Used to designate which memory location in the encoders to read out and encode. Note that this is actually the address of the event previous to the one requested by the trigger system. This is because the encoders are required to evaluate the previous RF bucket for 'hits' when encoding an event.

- ADDRESS VALID, Asynchronous , derived from the Master Timing controller signal . Used to validate Event Address.

- EVENT DATA, Synchronized to 26.5 Mhz CLK3, validated by data valid, 8-bits from each encoder module. Not bussed.

- DATA VALID, Synchronized to 26.5 Mhz CLK3, validates the event data, 1-bit from each encoder module. Not bussed.

- SYNC, This signal is received at the front panel, then delayed by the total of CLK1 delay plus CLK2 delay, then sent to the encoder modules. It is used to verify that all encoder module write counters are at the same count (zero) at sync time. Active low.

- SYNC ERROR, Asynchronous bussed signal from the encoder modules, signifies that one or more of the encoders is out of sync with the system.

1.4. Power Requirements

+5 Volts @ 9 Amps

-5.2 Volts @ 3 Amps

-2 Volts @ 2 Amps

1.4.1. Control and Monitoring Requirements

No requirement exists for a special control or monitoring of the power supply to the sequencer module. The normal protection provided by the FASTBUS crate enviremental system is sufficient.

1.5. Cooling Requirements

The sequencer represents a heat load of 65 watts which must be absorbed and carried away by the FASTBUS cooling system.

1.6 Sequencer Module SSD Auxiliary Backplane Pin List

C01-H53MHZ,Ø1 Clock, Slot 25,23,	B01-Reset	A01-H53MHZ,Ø1 Clock,Slot 11,
C02-L53MHZ,Ø1 Clock, 21,19,17,15	B02-GND	A02-L53MHZ,Ø1 Clock,9,7,5,3,1
C03-Hit Data 0, Slot 24	B03-Fiber Error	A03-Hit Data 0, Slot 2
C04-Hit Data 1, Slot 24	B04-GND	A04-Hit Data 1, Slot 2
C05-Hit Data 2, Slot 24	B05-Fiber Wait	A05-Hit Data 2, Slot 2
C06-Hit Data 3, Slot 24	B06-GND	A06-Hit Data 3, Slot 2
C07-Hit Data 4, Slot 24	B07-Fiber Clock	A07-Hit Data 4, Slot 2
C08-HIt Data 5, Slot 24	B08-GND	A08-Hit Data 5, Slot 2
C09-Hit Data 6, Slot 24	B09-Sync	A09-Hit Data 6, Slot 2
C10-Hit Data 7, Slot 24	B10-Sync Err	A10-Hit Data 7, Slot 2
C11-Data Valid, Slot 24	B11-GND	A11-Data Valid, Slot 2
C12-VCC, +5.0 Volts	B12-Fiber D00	A12-VEE, -5.2 Volts
C13-Hit Data 0, Slot 20	B13-GND	A13-Hit Data 0, Slot 6
C14-Hit Data 1, Slot 20	B14-Fiber D01	A14-Hit Data 1, Slot 6
C15-Hit Data 2, Slot 20	B15-GND	A15-Hit Data 2, Slot 6
C16-Hit Data 3, Slot 20	B16-Fiber D02	A16-Hit Data 3, Slot 6
C17-Hit Data 4, Slot 20	B17-GND	A17-Hit Data 4, Slot 6
C18-HIt Data 5, Slot 20	B18-Fiber D03	A18-Hit Data 5, Slot 6
C19-Hit Data 6, Slot 20	B19-GND	A19-Hit Data 6, Slot 6
C20-Hit Data 7, Slot 20	B20-Fiber D04	A20-Hit Data 7, Slot 6
C21-Data Valid, Slot 20	B21-GND	A21-Data Valid, Slot 6
C22-GND	B22-Fiber D05	A22-GND
C23-Hit Data 0, Slot 16	B23-GND	A23-Hit Data 0, Slot 10
C24-Hit Data 1, Slot 16	B24-Fiber D06	A24-Hit Data 1, Slot 10
C25-Hit Data 2, Slot 16	B25-GND	A25-Hit Data 2, Slot 10
C26-Hit Data 3, Slot 16	B26-Fiber D07	A26-Hit Data 3, Slot 10
C27-Hit Data 4, Slot 16	B27-GND	A27-Hit Data 4, Slot 10
C28-HIt Data 5, Slot 16	B28-Fiber D08	A28-Hit Data 5, Slot 10
C29-Hit Data 6, Slot 16	B29-GND	A29-Hit Data 6, Slot 10
C30-Hit Data 7, Slot 16	B30-Fiber D09	A30-Hit Data 7, Slot 10
C31-Data Valid, Slot 16	B31-GND	A31-Data Valid, Slot 10
C32-GND	B32-Fiber D10	A32-VEE, -5.2 Volts
C33-Left 26 MHZ Clock	B33-GND	A33-Right 26 MHZ Clock
C34-Hit Data 0, Slot 14	B34-Fiber D11	A34-Hit Data 0, Slot 12
C35-Hit Data 1, Slot 14	B35-GND	A35-Hit Data 1, Slot 12
C36-Hit Data 2, Slot 14	B36-Fiber D12	A36-Hit Data 2, Slot 12
C37-Hit Data 3, Slot 14	B37-GND	A37-Hit Data 3, Slot 12
C38-Hit Data 4, Slot 14	B38-Fiber D13	A38-Hit Data 4, Slot 12
C39-HIt Data 5, Slot 14	B39-GND	A39-Hit Data 5, Slot 12
C40-Hit Data 6, Slot 14	B40-Fiber D14	A40-Hit Data 6, Slot 12
C41-Hit Data 7, Slot 14	B41-GND	A41-Hit Data 7, Slot 12
C42-Data Valid, Slot 14	B42-Fiber D15	A42-Data Valid, Slot 12
C43-GND	B43-GND	A43-VCC, +5.0 Volts
C44-Hit Data 0, Slot 18	B44-Fiber Mux Enable	A44-Hit Data 0, Slot 8
C45-Hit Data 1, Slot 18	B45-Fiber User 2	A45-Hit Data 1, Slot 8
C46-Hit Data 2, Slot 18	B46-GND	A46-Hit Data 2, Slot 8
C47-Hit Data 3, Slot 18	B47-GND	A47-Hit Data 3, Slot 8
C48-Hit Data 4, Slot 18	B48-Event Address Valid	A48-Hit Data 4, Slot 8
C49-HIt Data 5, Slot 18	B49-Event Address Wrt En	A49-Hit Data 5, Slot 8
C50-Hit Data 6, Slot 18	B50-Event Address 0	A50-Hit Data 6, Slot 8
C51-Hit Data 7, Slot 18	B51-Event Address 1	A51-Hit Data 7, Slot 8
C52-Data Valid, Slot 18	B52-Event Address 2	A52-Data Valid, Slot 8
C53-VCC, +5.0 Volts	B53-Event Address 3	A53-GND
C54-Hit Data 0, Slot 22	B54-Event Address 4	A54-Hit Data 0, Slot 4
C55-Hit Data 1, Slot 22	B55-Event Address 5	A55-Hit Data 1, Slot 4
C56-Hit Data 2, Slot 22	B56-Event Address 6	A56-Hit Data 2, Slot 4
C57-Hit Data 3, Slot 22	B57-Event Address 7	A57-Hit Data 3, Slot 4
C58-Hit Data 4, Slot 22	B58-GND	A58-Hit Data 4, Slot 4
C59-HIt Data 5, Slot 22	B59-GND	A59-Hit Data 5, Slot 4
C60-Hit Data 6, Slot 22	B60-Fiber User 1	A60-Hit Data 6, Slot 4
C61-Hit Data 7, Slot 22	B61-Fiber User 0	A61-Hit Data 7, Slot 4
C62-Data Valid, Slot 22	B62-GND	A62-Data Valid, Slot 4
C63-VEE, -5.2 Volts	B63-Fiber Mux Control	A63-GND
C64-H53MHZ,Ø2 Clock, Slot 24,22,	B64-GND	A64-H53MHZ,Ø2 Clock,Slot 12,10,
C65-L53MHZ,Ø2 Clock, 20,18,16,14	B65-VTT, -2 Volts	A65-L53MHZ,Ø2 Clock 8,6,4,2

2. THEORY OF OPERATION AND OPERATING MODES

The sequencer module sits in the middle of the FASTBUS crate and communicates with the FASTBUS encoder modules via a special auxiliary backplane. Front panel connectors on the module connect via cables to the master controller to provide system clock, trigger commands, system RESET, errors and etc. The sequencer sends its data out via a rear mounted auxiliary card. Data may also be read via FASTBUS transfers.

2.1. Basic Operation

At power up and system reset time (i.e., the reset generated by the Master Timing Controller) the Sequencer Module will go into the initialization mode. In this mode CLK1 is not being driven onto the Auxiliary Backplane. The control logic waits for the Master Timing Controller to assert Write_Enable and then waits for the Master Timing Controller to assert Sync. When Sync is received the Sequencer will start driving CLK1 onto the Auxiliary Backplane. At this point the module is in normal running mode.

On receipt of an event address from the Master Timing Controller, the sequencer broadcasts it to the encoder modules via the special auxiliary backplane. Each encoder module extracts from memory, the 128 channels of hit information from that event and encodes it into an ordered list of 8-bit binary numbers. Each number in the hit list is comprised of a 7-bit binary number identifying the strip which had been driven above threshold by an ionizing particle and 1-bit to indicate if that microstrip had also registered a hit during the previous bucket (thus indicating that the current hit might actually be a residual from that bucket). All encoders do their encoding concurrently and send the data over individual data paths on the auxiliary backplane to FIFOs in the sequencer module.

As the sequencer begins receiving data from the encoder modules, it commences delivering it in order to its I/O ports. Beginning with the lowest numbered encoder FIFO, the data is transferred into the auxiliary card for transmission and into the FASTBUS buffer FIFO for readout. All the data from the 12 encoders are concatenated into a single ordered list. The ordering of the list is set by the encoder modules as lowest hit address to highest.

As each encoder FIFO produces an 'end of event token', the control logic moves on to the next. When the last FIFO has been emptied, the control logic adds the contents of the BLOCK COUNT/WORD COUNT and an IGNORE WORD if necessary to fill out the last 32-bit word in the data stream.

Referring to the block diagram, the sequencer consists of seven functional blocks.

2.1.1. Master Timing Controller Interface

- Receive 53 Mhz Clock. A function of the sequencer is to distribute clocks to the discriminator and encoder modules. A front panel connector on the sequencer receives a 53 Mhz clock from the master control module. This clock is exactly the same phase at all the sequencers. This clock has a fixed phase relationship to the signals arriving from the silicon microstrip detector. The sequencer derives two 53 Mhz clocks from this signal, CLK1 and CLK2. CLK1 is delayed from the input clock by a FASTBUS programmable amount and is driven to the discriminator modules via transmission lines on the auxilary backplane. CLK2 is delayed from CLK1 by a switch selectable amount and is driven to the encoder modules via transmission lines on the auxilary backplane. A divide-by-two version of CLK2 called CLK3 is bussed by the sequencer to the encoder modules to be used as a data readout clock.

- Receives system WRITE ENABLE signal from the Master Timing Controller.

- Receives SYNC signal from the Master Timing Controller.

- Receives RESET signal from the Master Timing Controller.

- Receives EVENT ADDRESSES and ADDRESS VALID from the Master Timing Controller.

- Sends 'sequencer ready' signal to the Master Timing Controller (SEQ_READY).

- Sends 'encoders ready' signal to the Master Timing Controller (EN_READY).

- Sends ERROR signal to the Master Timing Controller. This signal alerts to an encoder FIFO overflow, encoder out of sync and others to be added. FASTBUS can interrogate the Error Status Register to evaluate the error. Event size greater than 255 is not an error condition.

2.1.2. Encoder Interface

- Contains the FIFOs which receive data from each of the 12 encoder modules.
- Deposits 'end of event' token into each FIFO after all data has been received from the encoder.
- Sends RESET signal to the Encoder modules on command of the front panel signal.
- Sends WRITE ENABLE signal to the Encoder modules.
- Sends EVENT ADDRESSES and ADDRESS VALID to the Encoder modules.
- Sends SYNC to the Encoder modules delayed by CLK1 plus CLK2 delay.
- Receives SYNC ERROR from the Encoder modules.
- Receives EVENT DATA and DATA VALID from each Encoder module.

2.1.3. FASTBUS Interface

- Allows a FASTBUS master to read/write the control and status register.
- Allows a FASTBUS master to read/write the PLANE/ENCODER RAM.
- Allows a FASTBUS master to read/write the BLOCK COUNT/WORD COUNT.
- Allows a FASTBUS master to read/write the CLK1 delay value and read the CLK2 delay value.
- Allows a FASTBUS master to read the EVENT BUFFER FIFO.

2.1.4. Auxiliary Interface

The auxiliary interface is a high speed outlet for event data. Unless disabled by a CSR0 bit, the sequencer will drive data out this port directly, bypassing the need for software intervention to read out data over FASTBUS. This interface is designed so that it can drive an RS-485, ECLine or fiber optic auxiliary module. The auxiliary module provides the special interface requirements of the link. The signals provided by the sequencer to the fiber optic interface are listed below. The other types of interface modules ie. RS-485 and ECLine will have to be designed to utilize these same signals.

The auxiliary port interface includes the following signals. See the appendixes for the actual pin numbers.

- D0-D15. 16 multiplexed data lines to the auxiliary module.
- MUX ENABLE to the auxiliary module.
- MUX CONTROL to the auxiliary module.
- WAIT signal from the auxiliary module.
- CLOCK from the auxiliary module.
- ERROR signal from the auxiliary module.
- USER 1- USER 3. Three user defined signals.

2.1.5. Block/Word Counters

The BLOCK COUNT/WORD COUNT is a 16-bit entity which is normally used as the last valid word of a concatenated event. It consists of a 4-bit type code, a 4-bit counter for use as a BLOCK COUNT and an 8-bit counter for use as a WORD COUNT. The 4-bit type code is a binary '1000' placed in bit position 15-12 to identify the data as being the BLOCK COUNT/WORD COUNT of an event as opposed to data. The 4-bit BLOCK COUNT is intended to be used as a processed event identifier to provide parallel synchronization among the 12 FASTBUS crates. The 8-bit WORD COUNT is intended to be used to count the number of 'hits' in each event. The counters are cleared and incremented individually by the control logic, but their combined contents are always output as a 16-bit word. The counters may be written and read via FASTBUS for diagnostic purposes.

2.1.6. FASTBUS Event FIFO

FASTBUS output will be via a single event FIFO (32 X 512). Unless disabled by a CSR0 bit, a full event including LAST WORD and IGNORE WORD if necessary will be loaded into this FIFO as it is being sent out the auxiliary port. Output to FASTBUS will be in 32 bit words. A 32 bit word count will be generated and inserted at the end of the event record exactly as is done for the auxiliary port. If the FASTBUS port is being used for readout, the sequencer will not start to read a second event out through the auxiliary port until the event in the FASTBUS FIFO has been completely read out. A CSR0 bit will disable the FASTBUS event FIFO to allow data output via the auxiliary port without regard to the status of the event FIFO.

2.1.7. Control Logic

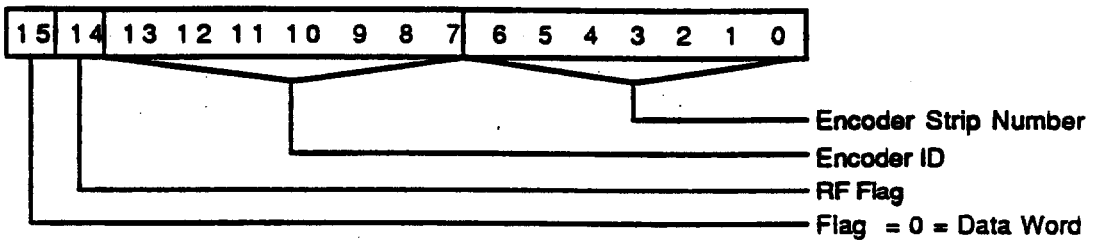
The control logic controls the input fifos, block and word counters, data identifier RAM, auxilliary port interface FIFO and FASTBUS event FIFO. This logic is implemented with programmable array logic devices (PALs).

2.2. Addressing Modes

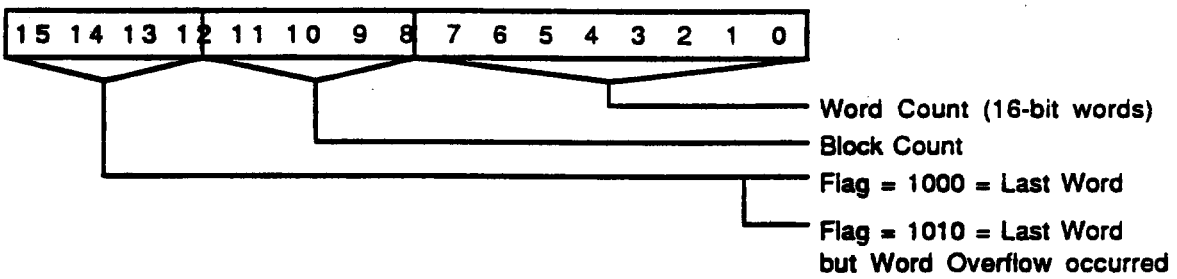
2.2.1. Data Transfer Description and Rates

The sequencer is designed to transfer silicon strip detector data over the auxiliary fiber optic link at peak instantaneous rates approaching 80 nsec per 32-bit word. The sequencer is designed to support block transfer data reads from its FASTBUS event buffer at peak instantanaeous rates approaching 150 nsec per 32-bit word. Whether data is output via the auxiliary port or read via the FASTBUS slave interface, it will always be in 32-bit words and each 32-bit word will always be comprised of two 16-bit words. The two 16-bit words words will each be of one of three types. The three types are illustrated below:

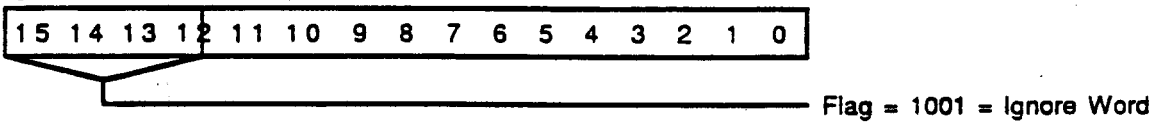
Data Word Format



Last Word Format



Ignore Word Format

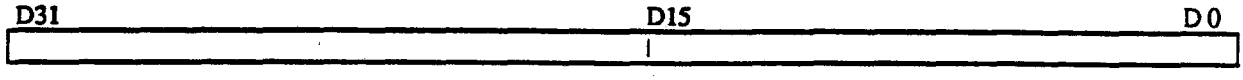


2.2.2. Internal Control, Status Registers, and Bit Descriptions

There are several registers on the Sequencer that are accessible via FASTBUS for initialization, performance monitoring, or diagnostics. These registers are described in the following subsections.

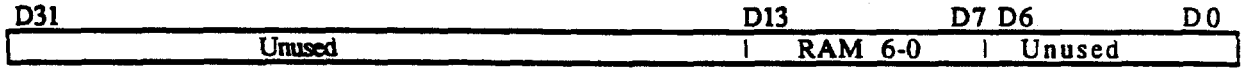
2.2.2.1. FASTBUS mandatory CSR0

CSR Space Hex Address		Description
0000 0000	CSR 0	Bit descriptions:
	Read , Write	
	Bit <6>,<22>	Enable/Disable sequencer loading of FASTBUS event FIFO.
	Bit <7>,<23>	Enable/Disable sequencer loading of auxiliary interface FIFO.
	Bit <8>,<24>	Enable/Disable overflow truncation.
	Bit ,<30>	Reset sequencer.
	Bits<31:16>,	Manufacturer's ID and device type.



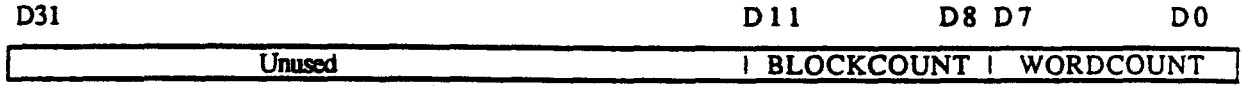
2.2.2.2. Read/Write The PLANE/ENCODER RAM

CSR Space Hex Address	Description
C000 0000 - C000 000B	There are 16 RAM locations accessible by FASTBUS of which 0 thru 11 are used to hold the 7-bit plane/encoder identifying data. 12 locations are required, one for each encoder FIFO. Although FASTBUS read/writes utilized 32-bits, only 7-bits are used for this operation and the rest are undefined. This memory may by written and read via FASTBUS not only for setting up the module but for diagnostics as well. This is only a 7-bit RAM. The MSB will always read '0' because this is a tag that differentiates DATA WORDs from LAST WORD or IGNORE WORD in the data stream.



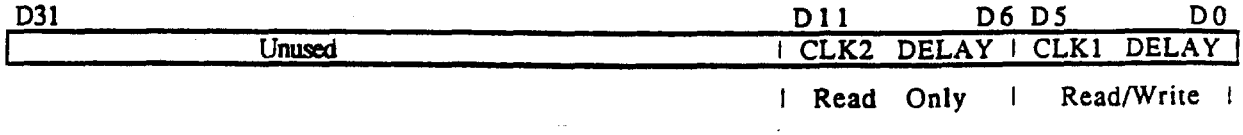
2.2.2.3. Read/Write The BLOCK COUNT/WORD COUNT

CSR Space Hex Address	Description
C000 0010	The BLOCK COUNT/WORD COUNT counters may be written and read via FASTBUS for diagnostic purposes.



2.2.2.4. Read/Write The CLK1 Delay Value, Read The CLK2 Delay Value

CSR Space Hex Address	Description
C000 0011	The CLK1 delay value is FASTBUS read/writeable, 0.5 ns per step, 64 steps. The 53 Mhz clock received via the front panel is delayed by this value to produce CLK1. This value is also added to the switch settable delay for CLK2 to produce the total delay value for the SYNC signal. The CLK2 delay value is set by hardware DIP switches on the card itself. The setting of those switches are read back in bits D11 through D6.



2.2.2.5. Read-Only Error Status Register

CSR Space Hex Address	Description
C000 0012	FASTBUS readable, bit D0 is set if a Sync Error has been detected by a Delay/Encoder. Bits D1 through D12 correspond to encoder FIFO channels 1 through 12. If one of these bits is set then its corresponding FIFO has overflowed.

D31	D12	D1	D0
Unused	Encoder Fifo Overflow Flags		Sync Err Flag

2.2.2.6. Read The EVENT BUFFER

In order to minimize overhead, the FASTBUS master may attempt to read from the event buffer even before it recieves a trigger and will be forced to WAIT only until data begins streaming into the FIFO. The master must not allow itself to time out while waiting for data. At the end of event processing, the sequencer will load the LAST WORD which contains the BLOCK COUNT/WORD COUNT; and an IGNORE WORD if necessary into the FIFO. After this word is read by the FASTBUS master, SS=2 would terminate the transfer.

The EVENT BUFFER is a 32-bit wide X 512 deep FIFO that holds a single event. The 32-bit word is made by packing two 'hits' which are normally 16-bits. To speed up FASTBUS transfers, this FIFO will be read out by a FASTBUS master using block transfer reads. There is a CSR0 bit to disable the event buffer if the event buffer is not to be used. This bit prevents the control logic from waiting for the buffer to be ready or strobing data into the buffer. This feature allows a FASTBUS master to sample events for histogramming or other reasons.

Data Space Hex Address	Description
0000 0000	Block transfer read of the event FIFO. Generate a primary address cycle to DATA space followed by block transfer reads. There is no secondary address cycle. The data transfers will be stalled by WAIT during the data cycle until event data begins streaming into the FIFO. The overhead involved with a 68020 powered FASTBUS Smart Crate Controller is as follows: 1) 300 nsec for the move instruction that executes a FASTBUS primary address cycle. 2) 300 nsec for the move instruction that starts the execution of a block transfer. 3) About 150 nsec per 32-bit word of data during the block transfer. The overhead from 1 and 2 is not incurred if the master executes the primary address cycle before a trigger is even received. The last word in the FIFO will have the 'LAST WORD' flag, the BLOCK COUNT and the WORD COUNT. If the number of words is odd, the last word will end up in the LS byte position of the 32-bit longword and an IGNORE WORD with a binary flag of '1100' will end up in the MS word. If the number of 32-bit words in the FIFO is even, the LAST WORD will end up in the MS word of the last 32-bit longword.

Data words:

D31	D15	D0
n+1 Data Word		n Data Word

Last word (if odd number of words):

D31	D15	D0
Ignore Word		Last Word

Last word (if even number of words):

D31	D15	D0
Last Word		n+x Data Word

2.2.3. Error Responses

The ERROR signal is sent to the Master Timing Controller when a Sync Error is detected of when and encoder FIFO overflows. The system software can then read the Error Status Register to determine specifically which of the encoder FIFOs overflowed or if a Sync Error had been detected by this sequencer.

2.3 Overview of the Sequencer Data Pipeline Operation

The twelve channel Delay/Encoder interface circuit control logic idles until it receives an Address Valid (ADVAL) pulse from the MTC. The Sequencer synchronizes the ADVAL signal and passes it on to all twelve Delay Encoder Modules. The Sequencer also uses ADVAL to start the Sequencer's Delay Encoder Timeout Counter. Upon receiving the ADVAL signal each Delay Encoder Module, that has data to send, will place an 8 bit data word on it's private Hit Data lines and activate it's own Hit Data Valid (DVAL) signal. There are twelve Encoder Fifo Circuits on the Sequencer which correspond to the twelve Delay Encoders. Encoder Fifo Circuits which receive a DVAL from their corresponding Delay Encoder will begin to write the 8 bit hit data words into their fifos (these will be referred to as the Encoder Fifos). On every positive transition of the CLK3 signal Delay Encoders with data to send will issue new 8 bit data words and the corresponding Sequencer Encoder Fifo Circuits will write this data into their fifos. When a Delay Encoder has no more data to send it deactivates DVAL. The Sequencer's Encoder Fifo Circuits see DVAL go away and they then write an End of Event Word (EOE word) into their fifos. The Encoder Fifos are 9 X 512. Bit 8 of the Encoder Fifo (the MSB) is high while hit data is being written and is low when we write the EOE word (I will refer to this bit from now on as the EOE tag). Encoder Fifo Circuits which do not receive DVAL signals before the Timeout Counter runs down (240ns after ADVAL received from the MTC will write an EOE tag into their fifos.

After the Timeout Counter runs down a trigger counter is incremented. Because the value contained in this trigger counter is no longer zero the Sequencer's Fifo Array Control Unit will begin a readout cycle by issuing a PRELOAD signal and decrementing the trigger counter. (Note that during a readout cycle the Sequencer will probably receive new triggers and the Sequencer's Encoder Fifo Circuits will be performing fifo writes concurrently with the readout cycle). The Preload signal causes all twelve Encoder Fifo Circuits to perform one data read from their fifos. Each Encoder Fifo Circuit places the 9 bit data word from this read in its output register. The Fifo Array Control Unit Examines the twelve bit vector comprised of the EOE tags from each Encoder Fifo Circuit. The Fifo Array Control Unit then reads out the first Encoder Fifo which has hit data. The data which is read out is placed in the P2 (pipeline stage 2) data register. When the Fifo Array Control Unit detects an EOE tag it immediately begins reading the next Encoder Fifo that contains hit data. There are no gaps in the pipeline data stream caused during this process. Channels which timed out are not read out.

Data from the P2 data register and a Delay Encoder ID from the Encoder ID RAM form a 16 bit word. The first 16 bit word constructed is written to the lower 16 bits of the 32 bit DMUX register. The next 16 bit data/ID word is written to the upper 16 bits of the DMUX register. 16 bit data/ID words continue to be alternately written to the lower and upper halves of the DMUX register. When the Fifo Array Control Unit detects that all event data words have been read out of the Encoder Fifos it issues the DONE signal to the Pipeline Control State Machine. This state machine will do one of two things. If there were an even number of data/ID words written to the DMUX register the state machine will cause a Last Word to be written to the lower 16 bits of the DMUX register and then it will cause an Ignore Word to be written to the upper 16 bits of the DMUX register. If there was an odd number of data/ID words written to the DMUX register, the state machine will cause a Last Word to be written to the upper 16 bits of the DMUX register. The last word is composed of the 8 bit word count, the 4 bit block count, and the 4 bit Last Word Identifier. The lower twelve bits of the Ignore word are undefined and the four remaining bits contain the Ignore Word ID Tag.

The 32 bit words constructed in the DMUX register will be written to either one (or both) of the 32 X 512 output fifos. One of the output fifos is called the FASTBUS Event Buffer; the other is known as the Auxiliary Event Buffer.

When the PIPEHOLD signal becomes active the data in the Sequencer Data Pipeline will be frozen in place and the current state of all state machines is preserved. The PIPEHOLD signal does not restrict the flow of data from the Delay Encoder Modules to the Sequencer Encoder Fifos. Only the sequencer's internal data pipeline is frozen. Below you will learn about situations which cause PIPEHOLD to become active.

If the FASTBUS Event Buffer is enabled but the Auxiliary Event Buffer is not enabled, DMUX register data will only be written to the FASTBUS Event Buffer. In this mode, once one complete event has been written to the FASTBUS Event Buffer, further writes are disabled until the previous event has been completely read out from the FASTBUS Event Buffer. In this mode we also freeze the Data Pipeline after each complete event has

been written into the FASTBUS Event Buffer. The pipeline remains frozen until the event in the FASTBUS Event Buffer has been completely read out. Note that, when in this mode, the pipeline will also be frozen if the FASTBUS event buffer becomes half full.

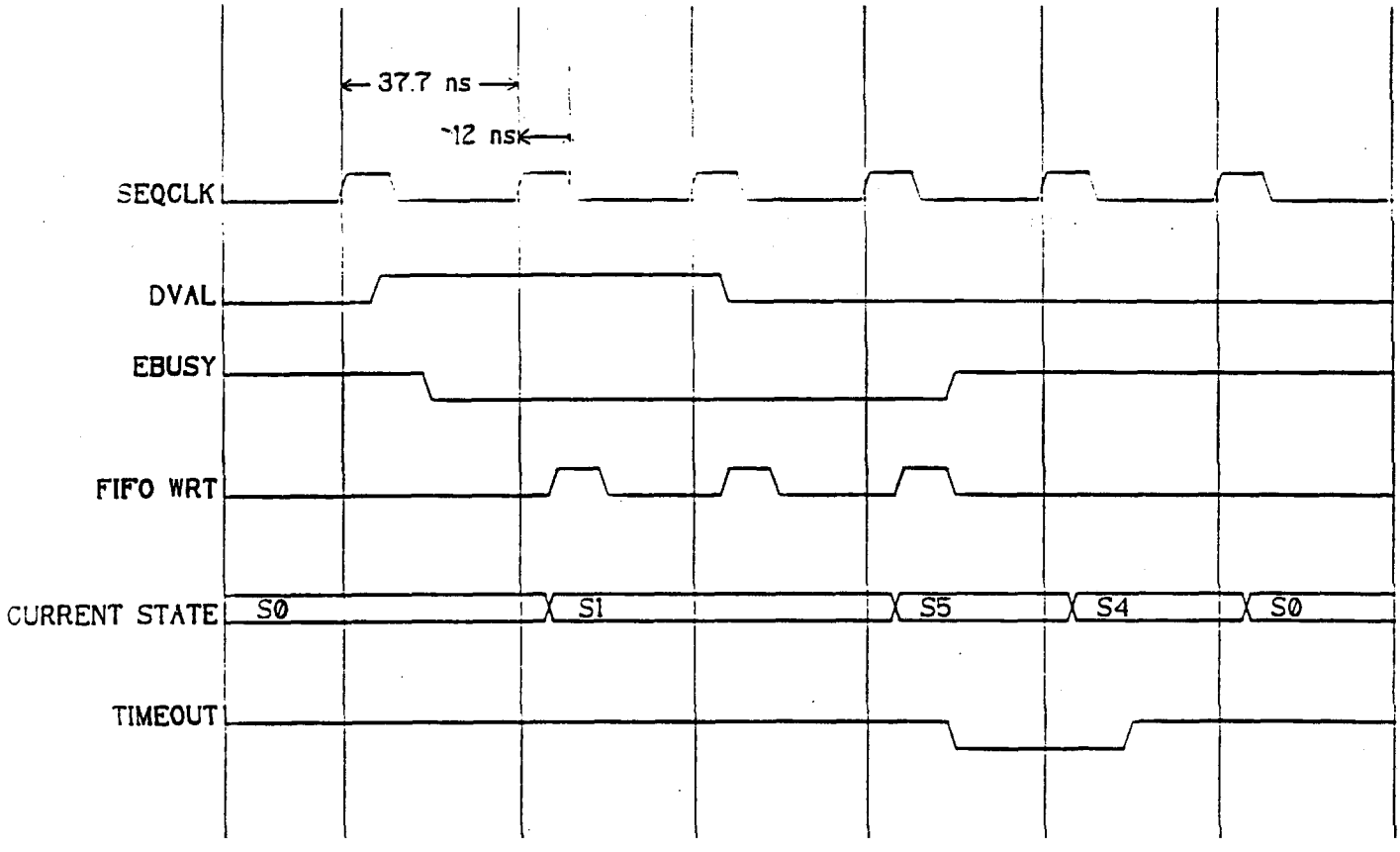
If both the FASTBUS and Auxiliary Event Buffers are enabled data from the DMUX register will be written to both event buffers. In this mode, once one complete event has been written to the FASTBUS Event Buffer, further writes to the FASTBUS Event Buffer are disabled until the previous event has been completely read out from the FASTBUS Event Buffer. Note that the presence of an event in the FASTBUS Event Buffer will NOT cause the pipeline to freeze. Data will continue to flow into the Auxiliary Event Buffer at full speed. This means that the FASTBUS Event Buffer will only receive a portion of the total events which travel through the pipeline. If the Auxiliary Event Buffer becomes half full the pipeline will freeze. It will remain frozen until the Auxiliary Event Buffer is no longer half full. In this mode, if the FASTBUS Event Buffer becomes half full the pipeline will be frozen. This should almost never happen since: 1) we only allow one event to exist in the FASTBUS Event Buffer and 2) most single events will contain less than 256 events. Thus Auxiliary data path performance will not be affected by the slower FASTBUS Data Path.

If only the Auxiliary Event Buffer is enabled, events will only be written to the Auxiliary Event Buffer. PIPEHOLD will only be activated in this mode when the Auxiliary Event Buffer becomes half full.

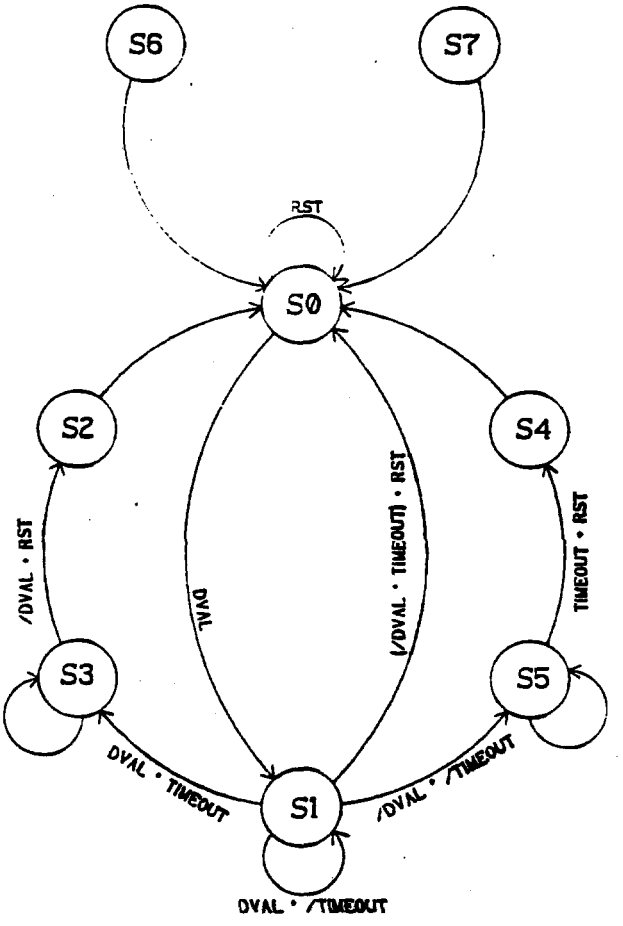
2.3.1 Encoder FIFO Circuits

There are twelve Encoder Fifo Circuits with one corresponding to each of the twelve Delay/Encoders. Each of these circuits contains an Encoder Fifo (75C01), a control pal and an output register section. The output register contains a data register pair (two 74AS574 8-bit registers and delay line) and a 74F74 1-bit register. The function of the control pal is to control the gating of the fifo read and fifo write signals and also to latch the occurrence of a FIFO full error condition. After Delayed Reset (DLYRST) is removed both the fifo write and fifo read signals go low. We use a delayed version of reset because fifo timing parameters require fifo reset to be inactive for 15ns before the read or write signals are dropped low. As shown in the timing diagrams below EBUSY becomes active after DVAL goes active and SEQCLK goes low. Whenever EBUSY is active, SEQCLK is allowed to flow out of the pal and clock data into the fifo. Once active EBUSY stays active as long as DVAL is active. When DVAL goes inactive the state machine in this pal will allow EBUSY to stay active for one more clock tick to allow an EOE tag to be written into the Fifo. After the EOE tag is written and SEQCLK has gone low we allow EBUSY to go inactive. By using SEQCLK as a qualifier for both the activation and deactivation of EBUSY we eliminate glitches on the fifo write line. Besides functioning as a gating control signal EBUSY, when active, is also used to remove the Encoder Ready signal which will be sent to the MTC. The operation of the control pal state machine is shown in the state diagram below. In State S0 the state machine waits for a DVAL signal from the Delay Encoder. State S0 is also the only state in which we allow an EOE tag to be written due to a time out. States S1 and S3 hold the fifo write gating signal (EBUSY) active until DVAL is inactive. If DVAL goes inactive before time out occurs the state machine enters S5 and waits for the time out signal to occur. This prevents us from writing 2 EOE tags into the fifo (one for the event data and one because time out occurred). States S2 and S4 are present to insure that all state transitions are 1-bit transitions. This will help to reduce noise that may occur because there are twelve of these circuits operating at the same time. States S6 and S7 are not used and are defined only for the purpose of trap state prevention. The operation of the fifo read gating control is similar to that of fifo write. Refer to the pal file ENCEN.ABL included with this document for more information on this chip.

Encoder FIFO Write Timing Diagram



Encoder FIFO Circuit State Machine



When an 8 bit event data word is read from the Encoder Fifo it is placed in the input register of the data register pair. The reason two data registers are used is to eliminate clock skew induced hold time violations when passing data between pipeline stages. In this circuit, and in all other pipeline stages, a write signal derived from the Sequencer's on board 25 Mhz clock writes data into the input register of the register pair. This signal delayed by 12ns is used to write the output register. This insures that data being sent to a subsequent pipeline stage will not change before that subsequent stage can "grab" the data. Bit 8, the MSB (the EOE Tag bit) of the 9 bit data word, which is read from the Encoder Fifo, is placed in the 74F74.

2.3.2 FIFO Array Control Unit

The Fifo Array Control Unit is comprised of clock distribution circuitry, the Delay Encoder Timeout Counter (ARTMO.ABL), the trigger counter (TRIGCNT.ABL), the array control state machine (ARLOAD.ABL), and readout channel selection logic (RDEN1.ABL, RDEN2.ABL, and ARIDAD.ABL).

The delay encoder timeout counter pal is a very simple state machine. This state machine sits in a state waiting for ADVAL. When ADVAL occurs it starts a binary count. After six rising edges of SEQCLK this counter enters the timeout state. As soon as the SEQCLK goes low the encoder timeout signal ENCOTMO is made active. ENCOTMO is sent to all Encoder Fifo Circuits whether they need it or not. After timeout is sent, this chip issues the TRINC signal to increment the trigger counter. During the timeout count this chip issues a signal called EBUSY13 which causes the sequencer to tell the MTC that the Encoders are not ready. We need this feature to account for the case where none of the Delay Encoders have data to send. We don't want the MTC to send the Sequencer another ADVAL until the current timeout count has finished.

The trigger counter pal's purpose is to tell the array control state machine whether or not the there are events in the Encoder Fifos to be read out. There are events to be read out if trigger counter's NOTEMP signal is active. The NOTEMP signal is active whenever the trigger count value is not equal to zero. The trigger counter is incremented if the increment signal (TRINC) is active and the decrement signal is not active. Note that the TRINC signal is issued after there is something in every encoder fifo (data or EOE tag) from the current trigger being encoded by the Delay Encoder. The trigger counter is decremented if the decrement signal (DEC) but not the increment signal (TRINC) is active. While the sequencer's data pipeline is processing an event, the trigger counter will allow up to 30 triggers to be received before activating the HF signal which deactivates the Sequencer Ready signal. If the MTC ignores the fact that the Sequencer Ready signal has gone away the trigger counter will be able to receive 33 more triggers before the ERROR signal is sent to the MTC.

If the pipeline is not frozen, the pipeline control state machine has completed the previous event readout cycle and the NOTEMP signal is active the array control state machine will issue the PRELOAD command to the Encoder Fifo Circuits. The array control state machine will then activate the event readout enable signal (EVENTEN) and keep it active until an active DONE signal is received from the readout channel selection logic. When DONE is received if no errors (full encoder fifo or trigger counter overflow) have occurred the array control state machine will return to the hold state and wait for the NOTEMP signal to become active again. If, when DONE is received, an error has occurred the array control state machine will enter the ERROR state and remain there until the sequencer is reset.

The readout channel selection logic decodes the 12-bit EOE tag vector and determines which of the twelve Encoder Fifo Circuits (channels) will be read out. The readout channel selection logic also uses the EOE tag vector to produce a Encoder ID address.

2.3.3 Data Word Construction Unit

The Data Word Construction Unit is comprised of the P2 register (two 74AS574 registers), the Block Counter (74F569), the Word Counter (74AS867), counter control logic (BWCOUNTC.ABL), the Encoder ID Ram (CY7C190), the Flag Register (74AS244), the Pipeline Control State Machine (BWSTATEM.ABL), the Pipeline Control State Decoder (BWSTDCD.ABL), write signal gating control (BWGATEC.ABL and BWHOLD.ABL), the DMUX Register (eight 74AS574 registers) and as always, miscellaneous glue logic (BWRAMCON.ABL).

The P2 Register is a back to back register pair comprised of two 74AS574 8-bit registers. The P2 input register receives data over the 8 bit EFDATA bus from any one of the Encoder Fifo Circuits. The P2 output register, when enabled, places its data on the 16 bit DATA bus. Note that after flowing through the P2 Register bits 1-7 of the EFDATA bus are transferred to bits 0-6 of the DATA bus. Bit 0 (the previous hit bit) of the EFDATA bus is transferred to bit 14 of the DATA bus.

The Block Counter is a 4-bit counter that counts the number of events (triggers) the sequencer has processed. The Word Counter is an 8-bit counter that counts the number of 8-bit data words contained in an event. Both the word and block counters can be written to and read from by FASTBUS. Care should be taken to not write to

the counters during data taking operations. **Programmers should avoid writing FF(Hex) to the word counter since this puts the counter into an overflow condition.**

The Pipeline Control State Machine will, upon sampling an active EVENTEN signal, enter the DATLO state. This causes the Pipeline State Decoder to enable the P2 output register and the ID Ram so they can place their contents on the DATA bus. While in this state the state machine allows the DMUX register gating circuitry (BWGATEC.ABL) to pass a write strobe (DMLOW) to the low half of the DMUX register. The next clock tick (if DONE has not occurred) will cause the state machine to enter the DATHI state. In this state the P2 data register and the ID Ram are still enabled. The gating circuitry will now, however, be allowed to pass a write strobe to the upper half of the DMUX register. The state machine will switch back and forth between the DATLO and DATHI states until it receives an active DONE signal or, if truncation is enabled, a word counter overflow occurs. If an active DONE signal is received and the state machine is in the DATHI state the state machine will enter the LASTLO state. The LASTLO state enables the Block Counter and Word Counter output registers for output onto the DATA bus. The LASTLO state also causes the Flag register to put a 4 bit Last Word Identifier on the data bus. These items are written into the low half of the DMUX Register. The next rising edge of the 25 Mhz clock causes the state machine to enter the IGNORHI state. In this state the Flag Register will place a 4 bit Ignore Word Identifier on the DATA bus. If the state machine is in the DATLO state when DONE occurs, the state machine will enter the LASTHI state and the Lastword (Block and Word Counts and Last Tag) will be written into the upper half of the DMUX register.

If event truncation is enabled, a word counter overflow (more than 255 data words in a single event) occurs and DONE is not active the state machine enters the TRUNCHI state and a Lastword with a special 4 bit truncation tag is written to the upper half of the DMUX register. After the Lastword and possibly the Ignore word are written to the DMUX register, the state machine enters the Flush state. It will stay in the FLUSH state until EVENTEN goes away. Under most conditions EVENTEN will have gone away before the state machine enters this state and the state machine will exit this state on the next clock tick. If the state machine enters this state because of a word counter overflow it stays in this state until the extra data from this overflow event has been flushed from the Encoder Fifos. After leaving the FLUSH state the state machine will always enter the SET COUNT state in which the Word Counter will be cleared and the Block counter will be incremented. The state machine leaves the SET COUNT state on the next clock tick and goes to the WAIT state towait for EVENTEN to occur.

The write pulse gating logic is designed to provide glitch free write pulses (at the correct time) to the P2 Register, the DMUX register (low and high halves), and to the Auxiliary and FASTBUS Event Buffers.

3. INPUT/OUTPUT SPECIFICATIONS

3.1. Communication Interfaces

3.1.1. Fiber Optic Auxiliary Port

The sequencer auxiliary port can be fitted with an auxiliary module which provides a high speed fiber optic link. Details about this auxiliary module is found in another document by the Detector Electronic Systems Group. Data delivered by the sequencer fiber optic port will have the same format as data delivered by the sequencer FASTBUS event buffer interface. However, the fiber optic link protocol requires some additional control operations to support handshaking, error reporting, etc.

There is a CSR0 bit which may be used by a FASTBUS master to disable the auxiliary port if the port is not to be used. This bit prevents the control logic from waiting on a port ready signal and from strobing data into the auxiliary port.

3.1.1.1. Communication Protocol

The sequencer communicates with the auxiliary card in the following fashion:

When the Sequencer has data in its Auxiliary interface FIFO the sequencer drives Auiliary output data lines 0-15 with the first 16-bit data word and asserts MUX ENABLE. The auxiliary card provides a clock signal to the sequencer and all transfers are synchronized with this clock. Subsequent 16-bit data words are transferred by alternating the level of the MUX CONTROL signal in order to tell the auxiliary card whether the lower 16 bits or the upper 16 bits of the 32 bit data word is currently being sent.

First and all subsequent normal data words (transferred 16 bits at a time, MUX CONTROL is toggled):

D31	D15	D0
n+1 Data Word		n Data Word

Last word (if odd number of words):

D31	D15	D0
Ignore Word	Last Word	

Last word (if even number of words):

D31	D15	D0
Last Word	n+x Data Word	

4. SYSTEM, MODULE, CIRCUIT, OR CHIP DIAGNOSTICS

4.1. Hardware

4.1.1. Sequencer Test Card

The SSD Sequencer Test Card (STC) is a FASTBUS card that is designed to reside in a Delay/Encoder slot of an SSD crate. The STC possesses a FASTBUS interface and can be controlled and read out by a FASTBUS master. In order to test the SSD Sequencer the STC simulates the Sequencer interface of the SSD Master Timing Controller (MTC) and the Sequencer interface of the SSD Delay/Encoder (D/E).

The twelve channels of the Sequencer can be individually tested by moving the STC to each of the twelve D/E slots and repeating the test procedure. However, provisions are included for a ribbon cable connector that can be used to control a twelve channel Auxiliary card which can send data to all twelve Sequencer channels.

The STC has a 1024 by 9 Event Data RAM which is used to store the data that will be sent to the Sequencer when a cycle is initiated. Although 128 is the largest number of hits that the D/E is capable of sending out, having 1024 pieces of data available allows us to test the Sequencer's ability to handle a FIFO overflow.

Locations addressable from FASTBUS:

C000 0000	MTC Event Address Offset register which stores the eight bit value of the Event Address Offset that will be issued to the Delay/Encoder. Read/write.	
C000 0001	Bits <0>, <16>	Assert/deassert the MTC Write_Enable signal.
C000 0001	Bit <1>	Assert the MTC Reset signal.
C000 0001	Bit <2>	Assert the STC Reset signal.
C000 0001	Bit <3>, <19>	Start a cycle by asserting Address_Valid.
C000 0002		Delay/Encoder simulator Event Address Offset latch. Read/Write.

0000 0000 to 0000 03FF in Data Space.	1024 RAM locations for Event Data, RF Flag, and an additional bit that indicates to the STC's circuit that the end of event has been reached. 1024 by 9 is the total.
0000 0400 to 0000 07FF in Data Space.	1024 RAM locations that reside on the twelve channel D/E test card.

RIBBON CABLE SIGNALS FOR THE TWELVE CHANNEL DELAY/ENCODER TEST CARD

EVENT_DATA(0:8)	Seven bit Event Data, one bit RF Flag, one bit Last Strip flag
ED_ADDR(0:9)	Address for the Event Data
DATA_STRB	Event Data Data Strobe
DATA_RD	Data Read/Write Control Signal
RUN_MODE	Asserted for running the test. Deasserted for writing or reading Event Data.
DTC_PRESENT	This signal will indicate to the STC that the twelve channel Delay/Encoder Test Card is present and connected to the STC via the ribbon cable.

RIBBON CABLE SIGNALS FOR THE AUXILIARY INTERFACE TEST CARD

AUX_DAT(0:15)	Auxiliary Data
AT_FIFO_READ	Auxiliary Test Card FIFO Read Signal
AT_FIFO_EMPTY	Auxiliary Test Card FIFO Empty
COM_STRB	Command Strobe
COM_RD	Command Read/Write Control Signal

COMMAND(0:3)	Command Bits
ATC_PRESENT	This signal indicates to the STC that the Auxiliary Interface Test Card is present and connected to the STC via the ribbon cable.

4.1.2. Operating Instructions

Load the Event Data RAM into the STC via FASTBUS.

Issue an MTC RESET with a FASTBUS write to the STC.

Generate the SYNC signal with a FASTBUS write to the STC.

Assert WRITE_ENABLE with a FASTBUS write to the STC.

Output an Event Address Offset with a FASTBUS write to the STC.

Assert the MTC ADDRESS_VALID a FASTBUS write to with the STC.

The Delay/Encoder emulator circuit should then transfer data to the Sequencer.

Read the Event Data out of the FASTBUS Event Buffer FIFO with the FSCC and compare it with test data stored in the FSCC's memory.

Read the Event Data out of the Auxiliary Interface Test Card's FIFO via FASTBUS through the Test Control Card and compare the data to test data stored in the FSCC's memory.

Read the Delay/Encoder Event Offset Address Latch and compare the value to its test value stored in the FSCC's memory.

4.2. Software

4.2.1. Diagnostic Test Description

A detailed description of the Sequencer Diagnostic Software is shown in the Fermilab Program Note number PN 434 *Silicon Strip Detector System Single Board Diagnostic Tests.*

Software to test the funtionality to the sequencer includes the following tests:

Write and read the CLK1 delay value via FASTBUS.

Read the CLK2 delay value via FASTBUS.

Write and read the twelve encoder identification RAM locations from FASTBUS.

Write and read the block counter from FASTBUS.

Write and read the word counter from FASTBUS.

Write and read CSR 0 from FASTBUS.

Read the Error Status Register from FASTBUS.

Silicon Microstrip Detector Readout System

Sequencer Module

Bob DeMaat, Marc Larwill, Andy Romero

****

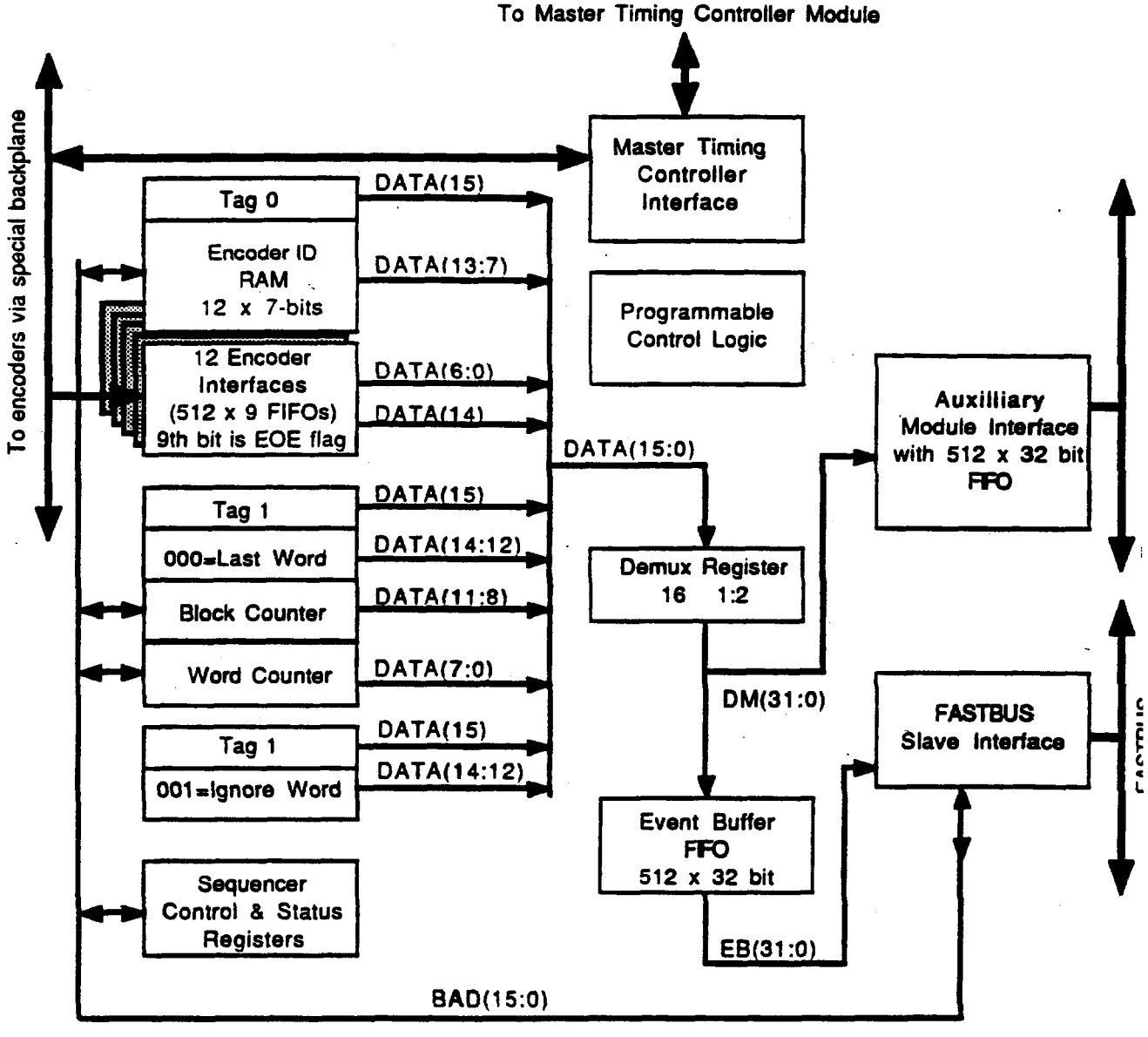
Preliminary Specification - Always check header date and use latest update - Direct all questions, comments and observations to Bob DeMaat.

****

1. GENERAL INFORMATION

1.1. Purpose

This document describes a device to be known as the "Silicon Strip Detector Readout System Sequencer Module" hereafter referred to as the sequencer. As one component of a larger system, this module will accept silicon strip detector data from the 12 encoder modules in its FASTBUS crate, combine the data and output it to other system components via FASTBUS or via an auxiliary link such as RS-485, ECLine or fiber optics. The general block diagram of the sequencer is as follows:

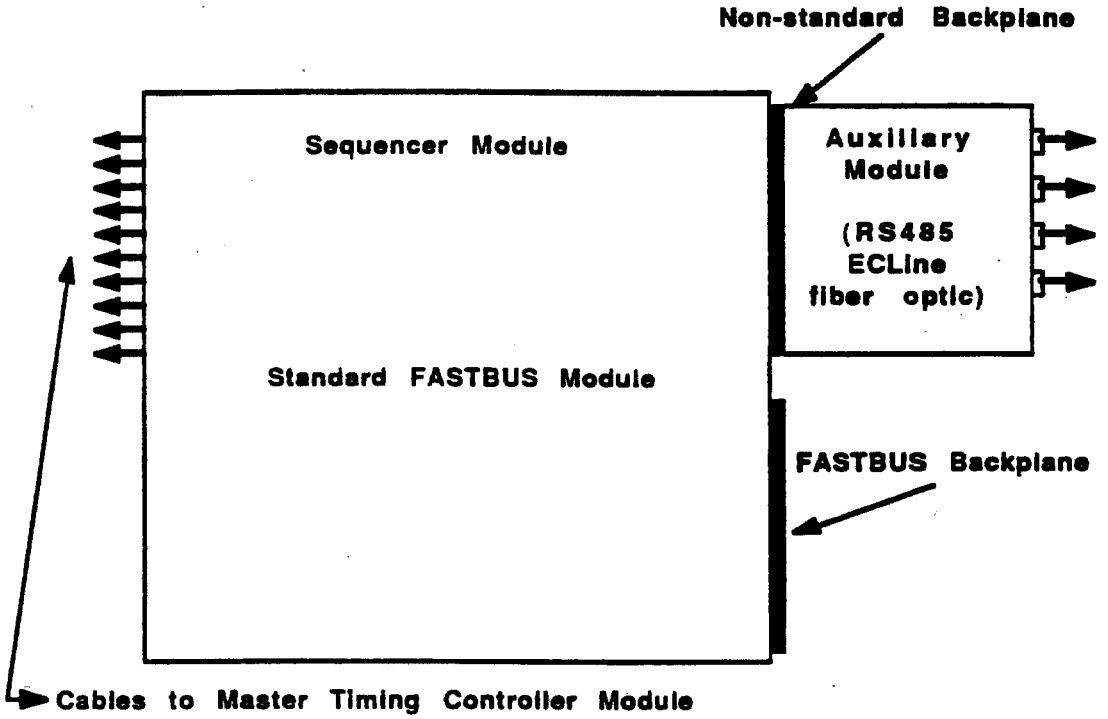


1.1.1. Standard Bus System Used

The sequencer is a single width FASTBUS module and includes FASTBUS slave capability. The FASTBUS slave is used for the initialization of some programmable parameters on the module and may be used to read out the encoded silicon strip detector data.

In addition to the standard FASTBUS interface, the sequencer uses a non-standard backplane on the FASTBUS auxiliary connector position to communicate with the encoder and postamp/discriminator modules. This backplane is optimized for high speed parallel transfers.

Some of these FASTBUS auxiliary connector pins are not used by the special backplane to communicate with the other modules. These pins are used instead by the sequencer to communicate with an I/O link driver such as RS-485, ECLine or fiber-optics. A special auxiliary interface module plugs into the back of the FASTBUS crate for this function. The figure describes this pictorially.

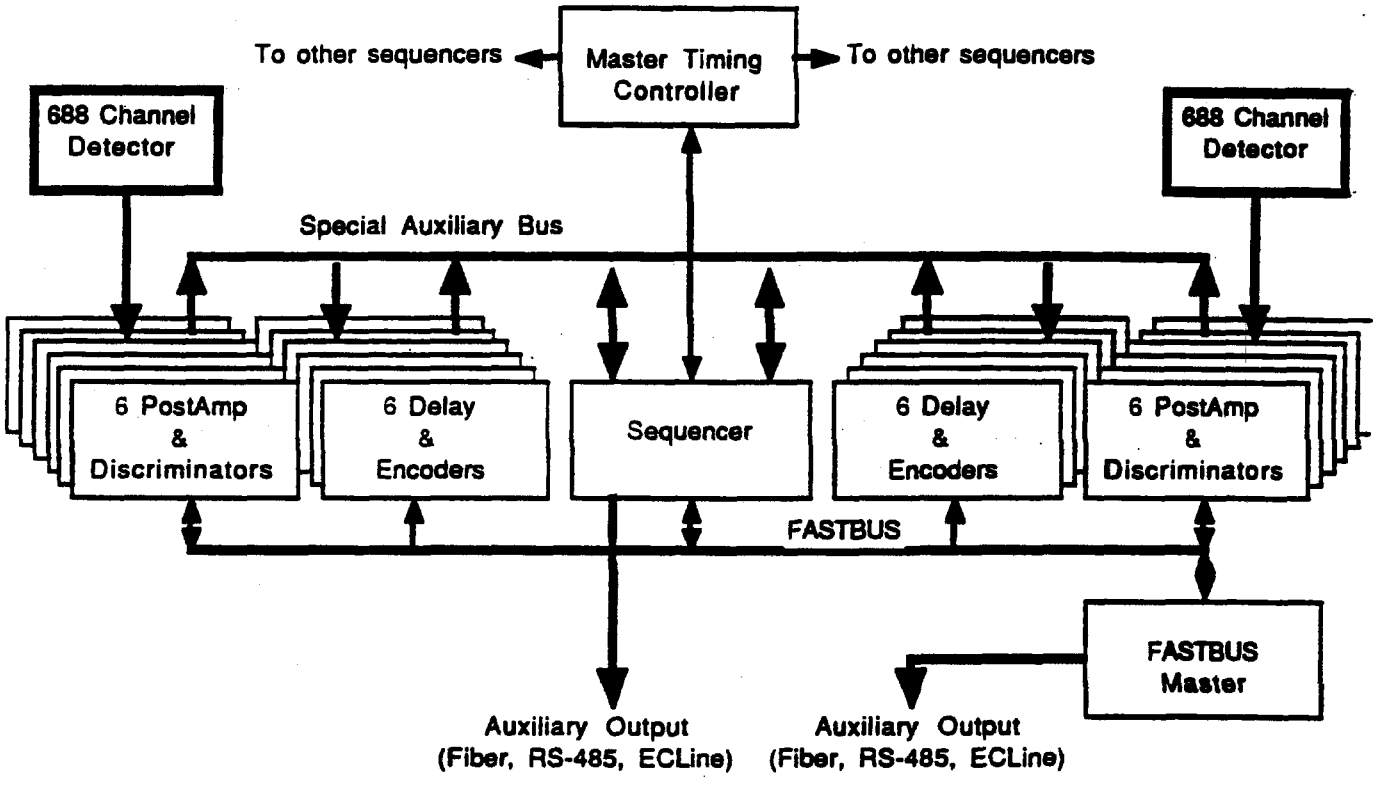


1.1.2. Number of Channels

Data comes into the sequencer from 12 encoder modules in the FASTBUS crate. The data is concatenated and may leave the sequencer via FASTBUS or auxiliary link (RS-485, ECLine or fiber optics).

1.2. Application

The sequencer module sits in the middle of the FASTBUS crate and communicates with the FASTBUS encoder modules via the special auxiliary backplane. Front panel connectors on the module connect via cables to the master controller to provide system clock, trigger commands, system RESET, errors and etc. The sequencer sends its data out via a rear mounted auxiliary card. Data may also be read via FASTBUS transfers. The sequencer may be used at Fermilab in experiment E771 and may be adaptable to other applications. In the E771 application, 24 planes of detector will be used requireing 12 crates of readout electronics. Each crate of electronics will be configured as pictured here:



1.3. Packaging

1.3.1. Module

The sequencer is implemented on a standard single width FASTBUS module which is 15.878" X 14.437" plus front panel. The module plugs into a FASTBUS crate which is then installed in an equipment rack along with the FASTBUS power and cooling system.

If the auxiliary port is used to output data, then a FASTBUS standard size auxiliary module is also used which contains logic and interface hardware to provide a high speed RS-485, ECLine or fiber optic data link. This auxiliary module plugs into the rear of the backplane at the sequencer module position in the slot provided for it by the FASTBUS crate.

1.3.2. Front and Rear Controls Connectors and Displays

1.3.2.1. CLK2 Delay Switch

The 53 Mhz clock received via a front panel connector is delayed by a programmable delay line and sent to the post amp/discriminator modules as a signal named CLK1. CLK1 is delayed by another programmable delay line and sent to the encoder modules as a signal named CLK2. The total amount of delay of CLK1 and CLK2 is also introduced on a signal called SYNC. The programmable delay line for CLK1 is programmed by FASTBUS, but the delay line for CLK2 is set by a 6-bit DIP switch. This switch provides a binary progressive delay from 0 to 31.5 nsec in .5 nsec increments.

1.3.2.2. Front Panel Displays

The sequencer module displays the following indicators of module operation and status on its front panel.

- FASTBUS SLAVE ACCESS; this LED indicator lights for a minimum of 100 msec each time a FASTBUS access to the sequencer occurs.

- SYNC ERROR, This LED indicator lights and remains lighted until RESET by FASTBUS any time an encoder module in the crate signals the sync error condition. Sync error is signalled by an encoder module if its write counter is not at the zero count when the sync signal is received from the Master Timing Controller via the sequencer.

- ENCODER FIFO OVERFLOW, This LED indicator lights and remains lighted until RESET by FASTBUS any time the encoder interface FIFOs are overfilled by data from the encoders. This is a system failure caused by the Master Timing Controller not using the SEQ_READY signal properly. When SEQ_READY is not asserted, the encoder data FIFOs are at least half full and the sequencer should not be given any new trigger addresses. At this time, there is room in the FIFOs to hold the data generated by a single trigger which may have been on its way to the sequencer before the Master Timing Controller received the SEQ_READY being deasserted. Any additional triggers sent by the Master Timing Controller after SEQ_READY is deasserted may generate enough data to overfill the FIFOs and generate the error.

- ENCODERS BUSY, This LED indicator is lighted by the logical 'OR' of the DATA VALID signals from each encoder in the crate. The signal is not 'stretched' or latched and normally occurs too fast to be seen. If this LED remains lighted, an encoder in the crate has failed and has effectively halted the system.

- SEQUENCER BUSY, This LED indicator is lighted by the logical 'OR' of the FIFO half-full signals from each encoder FIFO ion the sequencer module. The signal is not 'stretched' or latched and normally occurs too fast to be seen. If this LED remains lit, the sequencer is not processing the FIFO data out to FASTBUS and/or the auxiliary interface and has effectively halted the system. This may be caused by data not being removed from the FASTBUS event buffer or by the auxiliary link not being ready, or by PERMIT IN not being asserted.

- EVENT IN BUFFER, This LED indicator is lighted by the presence of data in the FASTBUS event FIFO. It will stay lit as long as that FIFO is not empty.

- EVENT SIZE OVERFLOW, this LED indicator is lighted by the occurance of an event which produces more that 255 'hits'. The sequencer does not 'know' this error has occurred until the data has already been received into its encoder FIFOs and is being counted as it is read out to FASTBUS and/or the auxiliary port. The occurrance of this error does not justify having to initialize the system yet a problem exists in how to dispose gracefully of the extra data. In normal operation the control logic will truncate an event at 255 'hits', properly insert the 'LAST WORD' into the data stream, and remove all extra data from the encoder FIFOs without FASTBUS intervention. Optionally, under the control of a bit in CSR0, the control logic will output the additional data while allowing the word counter to wrap around. This will be useful for system tests.

- +5 VOLT STATUS, this LED indicates that the board's +5 volt bus is powered up.

- -5.2 VOLT STATUS, this LED indicates that the board's -5.2 volt bus is powered up.

1.3.2.3. Front Panel Connectors

Signals marked '+' are 'party lined' in a single flat cable. One end connects to the Master Timing Controller (MTC) which terminates the signals. The sequencers attach at various places along the cable. The last sequencer module on the cable provides signal terminations.

- 53 Mhz CLOCK input, terminated NIM, from MTC.

+ WRITE ENABLE input, differential ECL, asynchronous to clock, from MTC. This signal is bussed to each encoder module over the auxiliary backplane. Enables encoders to accept hit data.

- SYNC input, terminated NIM, synchronous to clock, from MTC. This signal gets delayed by the same amount as CLK1 plus the same amount as CLK2 before being bussed to the encoder modules over the auxiliary backplane to test synchronization at each zero count.

+ RESET input, differential ECL, asynchronous to clock, resets control logic and fifos, bussed over the auxiliary backplane to the encoders where it resets write counters.

+ EVENT ADDRESS input, differential ECL, 8-bits, asynchronous to clock, bussed to encoder modules over the auxiliary backplane to designate a stored event. Note that this is actually the address of the event previous to the one requested by the trigger system. This is because the encoders are required to evaluate the previous RF bucket for 'hits' when encoding an event.

+ ADDRESS VALID input, differential ECL, asynchronous to clock, bussed to encoder modules over the auxiliary backplane to initiate encoding of a stored event.

+ SEQ_READY output, wire OR'd ECL, active low, signals that the sequencer has room in its fifos for events and thus can accept "read addresses".

+ ENC_READY output, wire OR'd ECL, active low, this is the sum of all encoder data valid signals. It signals to the master controller that the encoders are ready to accept another trigger.

+ ERROR output, wire OR'd ECL, active high, signals that the sequencer fifos have been overfilled by the encoders or that an encoder has lost synchronization. A fatal error requiring system RESET.

- PERMIT IN - Single ended TTL. A signal from the upstream module indicating that the sequencer may transmit. It's used to allow multiple sequencers to feed a party line.

- PERMIT OUT - Single ended TTL. A signal to a downstream module used to allow the next sequencer in a daisy chain to use the auxiliary party line if applicable.

1.3.2.4. Special Auxiliary Backplane Signals

- CLK1, delayed from the front panel 53 Mhz clock. Individually routed to each post amp/discrimminator module. The delay between the 53 Mhz clock received via the front panel, and this signal, is programmable via FASTBUS from 0 to 32 nsec in .5 nsec increments.

- CLK2, delayed from CLK1, individually routed to each encoder module. The delay between CLK1 and this signal is programmable via a 6-bit DIP switch from 0 to 32 nsec in .5 nsec increments.

- CLK3, 26.5 Mhz, CLK2 divided by two, bussed to all encoder modules for use as a data clock.

- RESET, Asynchronous, Bussed to all discriminators and encoder modules.

- WRITE ENABLE, Synchronized to 53 Mhz CLK2, bussed to all encoder modules, active low.

- EVENT ADDRESS, Asynchronous , 8-bits. Used to designate which memory location in the encoders to read out and encode. Note that this is actually the address of the event previous to the one requested by the trigger system. This is because the encoders are required to evaluate the previous RF bucket for 'hits' when encoding an event.

- ADDRESS VALID, Asynchronous , derived from the Master Timing controller signal . Used to validate Event Address.

- EVENT DATA, Synchronized to 26.5 Mhz CLK3, validated by data valid, 8-bits from each encoder module. Not bussed.

- DATA VALID, Synchronized to 26.5 Mhz CLK3, validates the event data, 1-bit from each encoder module. Not bussed.

- SYNC, This signal is received at the front panel, then delayed by the total of CLK1 delay plus CLK2 delay, then sent to the encoder modules. It is used to verify that all encoder module write counters are at the same count (zero) at sync time. Active low.

- SYNC ERROR, Asynchronous bussed signal from the encoder modules, signifies that one or more of the encoders is out of sync with the system.

1.4. Power Requirements

+5 Volts @ ?? Amps

-5.2 Volts @ ?? Amps

-2 Volts @ ?? Amps

1.4.1. Control and Monitoring Requirements

No requirement exists for a special control or monitoring of the power supply to the sequencer module. The normal protection provided by the FASTBUS crate enviremental system is sufficient.

1.5. Cooling Requirements

The sequencer represents a heat load of ?? watts which must be absorbed and carried away by the FASTBUS cooling system.

2. THEORY OF OPERATION AND OPERATING MODES

The sequencer module sits in the middle of the FASTBUS crate and communicates with the FASTBUS encoder modules via a special auxiliary backplane. Front panel connectors on the module connect via cables to the master controller to provide system clock, trigger commands, system RESET, errors and etc. The sequencer sends its data out via a rear mounted auxiliary card. Data may also be read via FASTBUS transfers.

2.1. Basic Operation

At power up and system reset time (i.e., the reset generated by the Master Timing Controller) the Sequencer Module will go into the initialization mode. In this mode CLK1 is not being driven onto the Auxiliary Backplane. The control logic waits for the Master Timing Controller to assert Write_Enable and then looks for the Master Timing Controller to assert Sync. When Sync is received the Sequencer will start driving CLK1 onto the Auxiliary Backplane. At this point the module is in normal running mode.

On receipt of an event address from the Master Timing Controller, the sequencer generates the event address-1 and broadcasts it to the encoder modules via the special auxiliary backplane. Each encoder module extracts from memory, the 128 channels of hit information from that event and encodes it into an ordered list of 8-bit binary numbers. Each number in the hit list is comprised of a 7-bit binary number identifying the strip which had been driven above threshold by an ionizing particle and 1-bit to indicate if that microstrip had also registered a hit during the previous bucket (thus indicating that the current hit might actually be a residual from that bucket). All encoders do their encoding concurrently and send the data over individual data paths on the auxiliary backplane to FIFOs in the sequencer module.

As the sequencer begins receiving data from the encoder modules, it commences delivering it in order to its I/O ports. Beginning with the lowest numbered encoder FIFO, the data is transferred into the auxiliary card for transmission and into the FASTBUS buffer FIFO for readout. All the data from the 12 encoders are concatenated into a single ordered list. The ordering of the list is set by the encoder modules as lowest hit address to highest.

As each encoder FIFO produces an 'end of event token', the control logic moves on to the next. When the last FIFO has been emptied, the control logic add the contents of the BLOCK COUNT/WORD COUNT and an IGNORE WORD if necessary to fill out the last 32-bit word in the data stream.

Referring to the block diagram, the sequencer consists of seven functional blocks.

2.1.1. Master Timing Controller Interface

- Receive 53 Mhz Clock. A function of the sequencer is to distribute clocks to the discriminator and encoder modules. A front panel connector on the sequencer receives a 53 Mhz clock from the master control module. This clock is exactly the same phase at all the sequencers. This clock has a fixed phase relationship to the signals arriving from the silicon microstrip detector. The sequencer derives two 53 Mhz clocks from this signal, CLK1 and CLK2. CLK1 is delayed from the input clock by a FASTBUS programmable amount and is driven to the discriminator modules via transmission lines on the auxilary backplane. CLK2 is delayed from CLK1 by a switch selectable amount and is driven to the encoder modules via transmission lines on the auxilary backplane. A divide-by-two version of CLK2 called CLK3 is bussed by the sequencer to the encoder modules to be used as a data readout clock.

- Receives system WRITE ENABLE signal from the Master Timing Controller.

- Receives SYNC signal from the Master Timing Controller.

- Receives RESET signal from the Master Timing Controller.

- Receives EVENT ADDRESSES and ADDRESS VALID from the Master Timing Controller.

- Sends 'sequencer ready' signal to the Master Timing Controller (SEQ_READY).

- Sends 'encoders ready' signal to the Master Timing Controller (EN_READY).

- Sends ERROR signal to the Master Timing Controller. This signal alerts to an encoder FIFO overflow, encoder out of sync and others to be added. FASTBUS can interrogate the Error Status Register to evaluate the error. Event size greater than 255 is not an error condition.

2.1.2. Encoder Interface

- Contains the FIFOs which receive data from each of the 12 encoder modules.
- Deposits 'end of event' token into each FIFO after all data has been received from the encoder.
- Sends RESET signal to the Encoder modules on command of the front panel signal.
- Sends WRITE ENABLE signal to the Encoder modules.
- Sends EVENT ADDRESSES and ADDRESS VALID to the Encoder modules.
- Sends SYNC to the Encoder modules delayed by CLK1 plus CLK2 delay.
- Receives SYNC ERROR from the Encoder modules.
- Receives EVENT DATA and DATA VALID from each Encoder module.

2.1.3. FASTBUS Interface

- Allows a FASTBUS master to read/write the control and status register.
- Allows a FASTBUS master to read/write the PLANE/ENCODER RAM.
- Allows a FASTBUS master to read/write the BLOCK COUNT/WORD COUNT.
- Allows a FASTBUS master to read/write the CLK1 delay value and read the CLK2 delay value.
- Allows a FASTBUS master to read the EVENT BUFFER FIFO.

2.1.4. Auxiliary Interface

The auxiliary interface is a high speed outlet for event data. Unless disabled by a CSR0 bit, the sequencer will drive data out this port directly, bypassing the need for software intervention to read out data over FASTBUS. This interface is designed so that it can drive an RS-485, ECLine or fiber optic auxiliary module. The auxiliary module provides the special interface requirements of the link. The signals provided by the sequencer to the fiber optic interface are listed below. The other types of interface modules ie. RS-485 and ECLine will have to be designed to utilize these same signals.

The auxiliary port interface includes the following signals. See the appendixes for the actual pin numbers.

- 40 data and control lines to the auxiliary module.
- Control/data strobe to the auxiliary module.
- Acknowledge from the auxiliary module.
- 6 status lines from the auxiliary module.
- Status Strobe from the auxiliary module.
- Link Error signal from the auxiliary module

The sequencer will be capable of initializing the data link by outputting a simple stream of control nibbles in response to a FASTBUS INITIALIZE command. The control/data lines from the auxiliary module will be used to indicate link errors. They will also be used to set/reset a latch indicating that the destination of the sequencer data has enough buffer space available to accept a maximum sized event. The sequencer will check this latch before sending an event out the auxiliary port.

2.1.5. Block/Word Counters

The BLOCK COUNT/WORD COUNT is a 16-bit entity which is normally used as the last valid word of a concatenated event. It consists of a 4-bit type code, a 4-bit counter for use as a BLOCK COUNT and an 8-bit counter for use as a WORD COUNT. The 4-bit type code is a binary '1000' placed in bit position 15-12 to identify the data as being the BLOCK COUNT/WORD COUNT of an event as opposed to data. The 4-bit BLOCK COUNT is intended to be used as a processed event identifier to provide parallel synchronization among the 12 FASTBUS crates. The 8-bit WORD COUNT is intended to be used to count the number of 'hits' in each event. The counters are cleared and incremented individually by the control logic, but their combined contents are always output as a 16-bit word. The counters may be written and read via FASTBUS for diagnostic purposes.

2.1.6. FASTBUS Event FIFO

FASTBUS output will be via a single event FIFO (32 X 512). Unless disabled by a CSR0 bit, a full event including LAST WORD and IGNORE WORD if necessary will be loaded into this FIFO as it is being sent out the auxiliary port. Output to FASTBUS will be in 32 bit words. A 32 bit word count will be generated and inserted at the end of the event record exactly as is done for the auxiliary port. If the FASTBUS port is being used for readout, the sequencer will not start to read a second event out through the auxiliary port until the event in the FASTBUS FIFO has been completely read out. A CSR0 bit will disable the FASTBUS event FIFO to allow data output via the auxiliary port without regard to the status of the event FIFO.

2.1.7. Control Logic

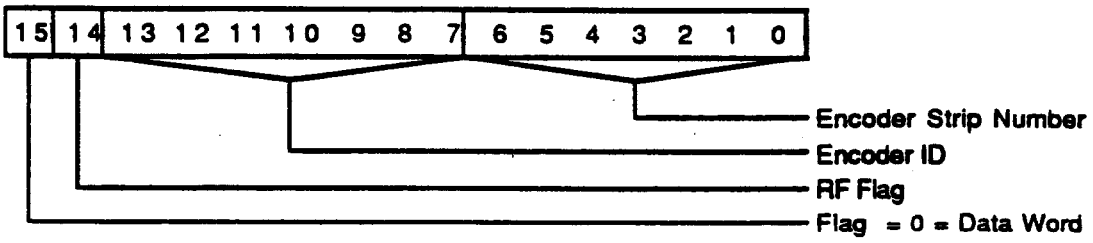
The control logic controls the input fifos, block and word counters, data identifier RAM, auxilliary port interface FIFO and FASTBUS event FIFO. This logic is implemented with programmable array logic devices (PALs).

2.2. Addressing Modes

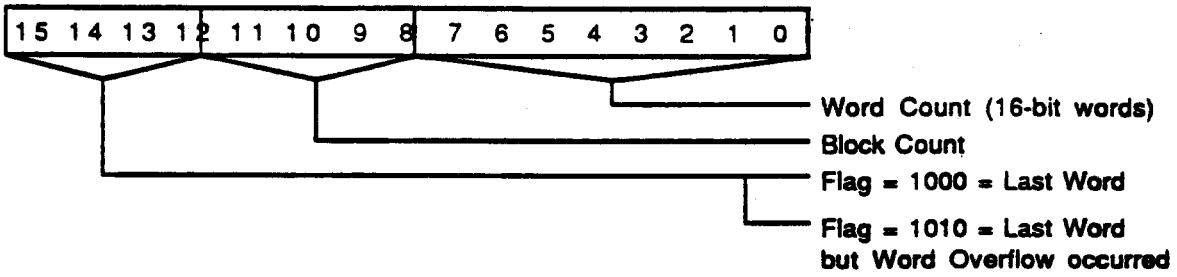
2.2.1. Data Transfer Description and Rates

The sequencer is designed to transfer silicon strip detector data over the auxiliary fiber optic link at peak instantaneous rates approaching 80 nsec per 32-bit word. The sequencer is designed to support block transfer data reads from its FASTBUS event buffer at peak instantanaeous rates approaching 150 nsec per 32-bit word. Whether data is output via the auxiliary port or read via the FASTBUS slave interface, it will always be in 32-bit words and each 32-bit word will always be comprised of two 16-bit words. The two 16-bit words words will each be of one of three types. The three types are illustrated below:

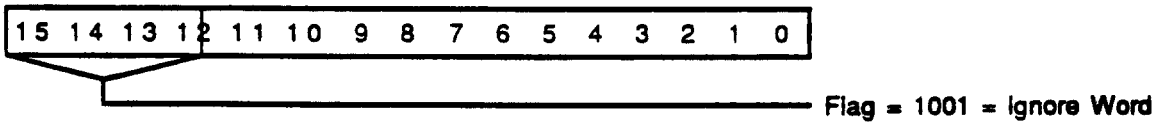
Data Word Format



Last Word Format



Ignore Word Format

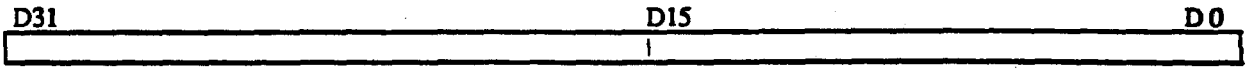


2.2.2. Internal Control, Status Registers, and Bit Descriptions

There are several registers on the Sequencer that are accessible via FASTBUS for initialization, performance monitoring, or diagnostics. These registers are described in the following subsections.

2.2.2.1. FASTBUS mandatory CSR0

CSR Space Hex Address		Description
0000 0000	CSR 0	Bit descriptions:
	Read , Write	
	Bit <6>,<22>	Disable/Enable sequencer loading of FASTBUS event FIFO.
	Bit <7>,<23>	Disable/Enable sequencer loading of auxiliary interface FIFO.
	Bit <8>,<24>	Disable/Enable overflow truncation.
	Bit ,<30>	Reset sequencer.
	Bits<31:16>,	Manufacturer's ID and device type.



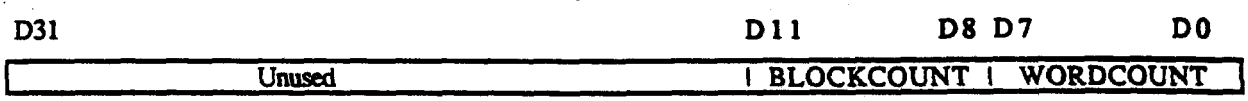
2.2.2.2. Read/Write The PLANE/ENCODER RAM

CSR Space Hex Address	Description
C000 0000 - C000 000B	There are 16 RAM locations accessible by FASTBUS of which 0 thru 11 are used to hold the 7-bit plane/encoder identifying data. 12 locations are

required, one for each encoder FIFO. Although FASTBUS read/writes utilized 32-bits, only 7-bits are used for this operation and the rest are undefined. This memory may by written and read via FASTBUS not only for setting up the module but for diagnostics as well. This is only a 7-bit RAM. The MSB will always read '0' because this is a tag that differentiates DATA WORDs from LAST WORD or IGNORE WORD in the data stream.

2.2.2.3. Read/Write The BLOCK COUNT/WORD COUNT

CSR Space Hex Address	Description
C000 0010	The BLOCK COUNT/WORD COUNT counters may be written and read via FASTBUS for diagnostic purposes.



2.2.2.4. Read/Write The CLK1 Delay Value, Read The CLK2 Delay Value

CSR Space Hex Address	Description
C000 0011	The CLK1 delay value is FASTBUS read/writeable, 0.5 ns per step, 64 steps. The 53 Mhz clock received via the front panel is delayed by this value to produce CLK1. This value is also added to the switch settable delay for CLK2 to produce the total delay value for the SYNC signal. The CLK2 delay value is set by hardware DIP switches on the card itself. The setting of those switches are read back in bits D11 through D6.

2.2.2.5. Read-Only Error Status Register

CSR Space Hex Address	Description
C000 0012	FASTBUS readable, bit D0 is set if a Sync Error has been detected by a Delay/Encoder. Bits D1 through D12 correspond to encoder FIFO channels 1 through 12. If one of these bits is set then its corresponding FIFO has overflowed.

D31	D12	D1	D0
Unused	Encoder Fifo Overflow Flags		Sync Err Flag

2.2.2.6. Read The EVENT BUFFER

In order to minimize overhead, the FASTBUS master may attempt to read from the event buffer even before it recieves a trigger and will be forced to WAIT only until data begins streaming into the FIFO. The master must not allow itself to time out while waiting for data. At the end of event processing, the sequencer will load the LAST WORD which contains the BLOCK COUNT/WORD COUNT; and an IGNORE WORD if necessary into the FIFO. After this word is read by the FASTBUS master, SS=2 would terminate the transfer.

The EVENT BUFFER is a 32-bit wide X 512 deep FIFO that holds a single event. The 32-bit word is made by packing two 'hits' which are normally 16-bits. To speed up FASTBUS transfers, this FIFO will be read out by a FASTBUS master using block transfer reads. There is a CSR0 bit to disable the event buffer if the event buffer is not to be used. This bit prevents the control logic from waiting for the buffer to be ready or strobing data into the buffer. This feature allows a FASTBUS master to sample events for histogramming or other reasons.

Data Space Hex Address	Description
0000 0000	Block transfer read of the event FIFO. Generate a primary address cycle to DATA space followed by block transfer reads. There is no secondary address cycle. The data transfers will be stalled by WAIT during the data cycle until event data begins streaming into the FIFO. The overhead involved with a 68020 powered FASTBUS Smart Crate Controller is as follows: 1) 300 nsec for the move instruction that executes a FASTBUS primary address cycle. 2) 300 nsec for the move instruction that starts the execution of a block transfer. 3) About 150 nsec per 32-bit word of data during the block transfer. The overhead from 1 and 2 is not incurred if the master executes the primary address cycle before a trigger is even received. The last word in the FIFO will have the 'LAST WORD' flag, the BLOCK COUNT and the WORD COUNT. If the number of words is odd, the last word will end up in the LS byte position of the 32-bit longword and an IGNORE WORD with a binary flag of '1100' will end up in the MS word. If the number of 32-bit words in the FIFO is even, the LAST WORD will end up in the MS word of the last 32-bit longword.

Data words:

D31	D15	D0
n+1 Data Word		n Data Word

Last word (if odd number of words):

D31	D15	D0
Ignore Word		Last Word

Last word (if even number of words):

D31	D15	D0
Last Word		n+x Data Word

2.2.3. Error Responses

The ERROR signal is sent to the Master Timing Controller when a Sync Error is detected of when and encoder FIFO overflows. The system software can then read the Error Status Register to determine specifically which of the encoder FIFOs overflowed or if a Sync Error had been detected by this sequencer.

2.2.4. Diagnostic Software

To be added

3. INPUT/OUTPUT SPECIFICATIONS

3.1. Communication Interfaces

3.1.1. Fiber Optic Auxiliary Port

The sequencer auxiliary port can be fitted with an auxiliary module which provides a high speed fiber optic link. Details about this auxiliary module is found in another document by the Detector Electronic Systems Group. Data delivered by the sequencer fiber optic port will have the same format as data delivered by the sequencer FASTBUS event buffer interface. However, the fiber optic link protocol requires some additional control operations to support handshaking, error reporting, etc.

There is a CSR0 bit which may be used by a FASTBUS master to disable the auxiliary port if the port is not to be used. This bit prevents the control logic from waiting on a port ready signal and from strobing data into the auxiliary port.

3.1.1.1. Communication Protocol

The sequencer communicates with the auxiliary card in the following fashion:

The sequencer will send an INITIALIZE command and then wait for the auxiliary module to return RECEIVED INITIALIZE status.

When the sequencer has data to transmit it issues INQUIRE commands until the auxiliary module returns READY status indicating that it can take the data.

The sequencer then transmits its data synchronized to the clock driven by the auxiliary card. as long as the auxiliary card is sending READY status.

If the auxiliary card sends NOT READY status while the sequencer is transmitting data the sequencer will revert to the state where it transmits INQUIRE commands until the auxiliary card again issues READY status and data transmission will pick up where it left off.

The sequencer will return to the INITIALIZE state when it runs out of data to transmit, when it times out waiting for status after issueing an INQUIRE command, or when it receives a LINK ERROR signal from the auxiliary card.

First and all subsequent normal data words:

D31	D15	D0
n+1 Data Word		n Data Word

Last word (if odd number of words):

D31	D15	D0
Ignore Word		Last Word

Last word (if even number of words):

D31	D15	D0
Last Word		n+x Data Word

4. SYSTEM SOFTWARE DESCRIPTION

4.1. Initialization Description Including Documented Code

Upon initialization, FASTBUS must be used to load the 'encoder identification RAMs', and to set the CLK1 delay. FASTBUS may also be used to write and read all internal registers for diagnostic purposes.

4.2. System Software

5. SYSTEM, MODULE, CIRCUIT, OR CHIP DIAGNOSTICS

5.1. Hardware

5.1.1. Special Test Modules and/or Test Setup Descriptions

5.1.2. Operating Instructions

5.2. Software

5.2.1. Diagnostic Test Description

Software to test the funtionality to the sequencer will include the following tests:

Write and read the CLK1 delay value via FASTBUS.

Read the CLK2 delay value via FASTBUS.

Write and read the twelve encoder identification RAM locations from FASTBUS.

Write and read the block counter from FASTBUS.

Write and read the word counter from FASTBUS.

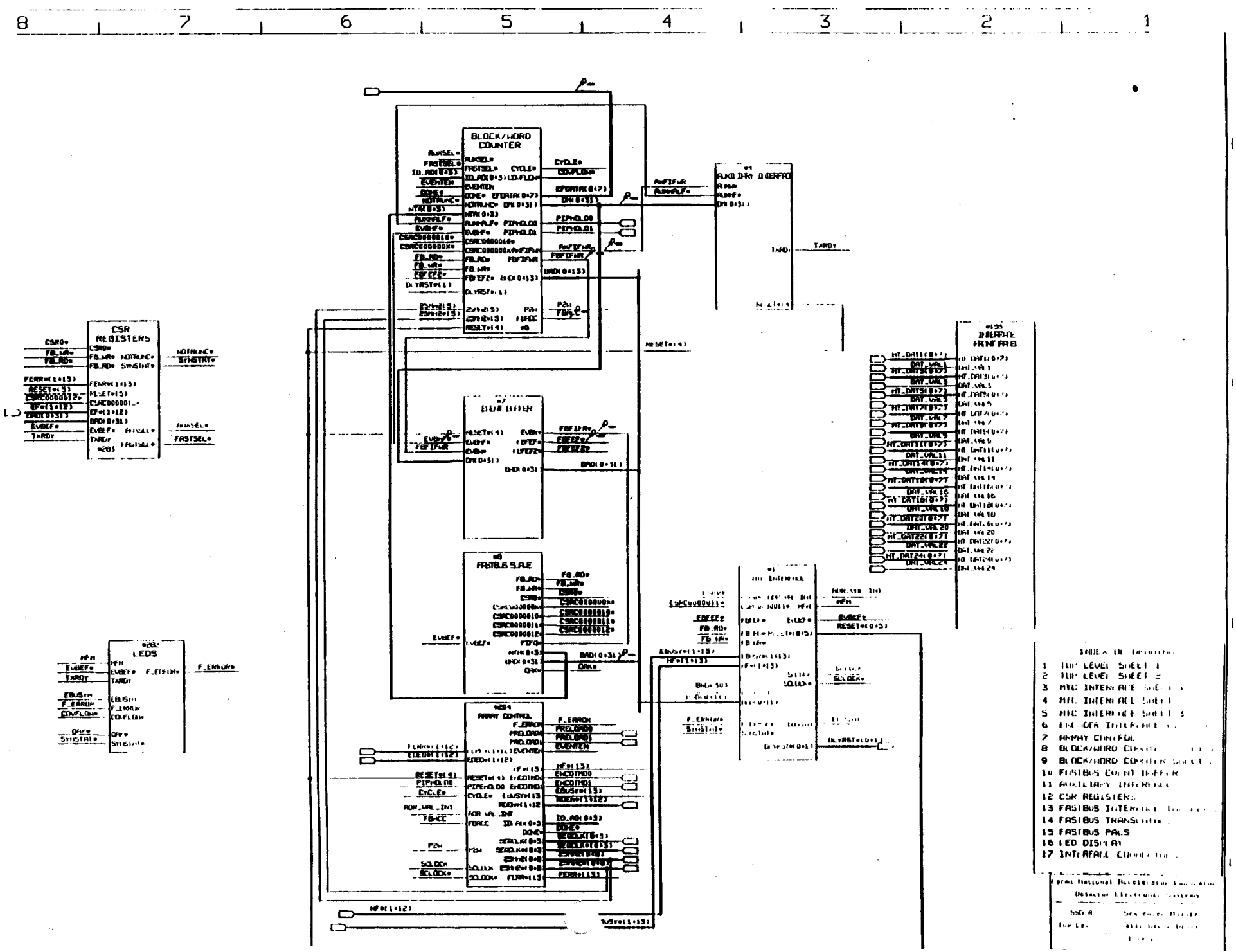
Write and read CSR 0 from FASTBUS.

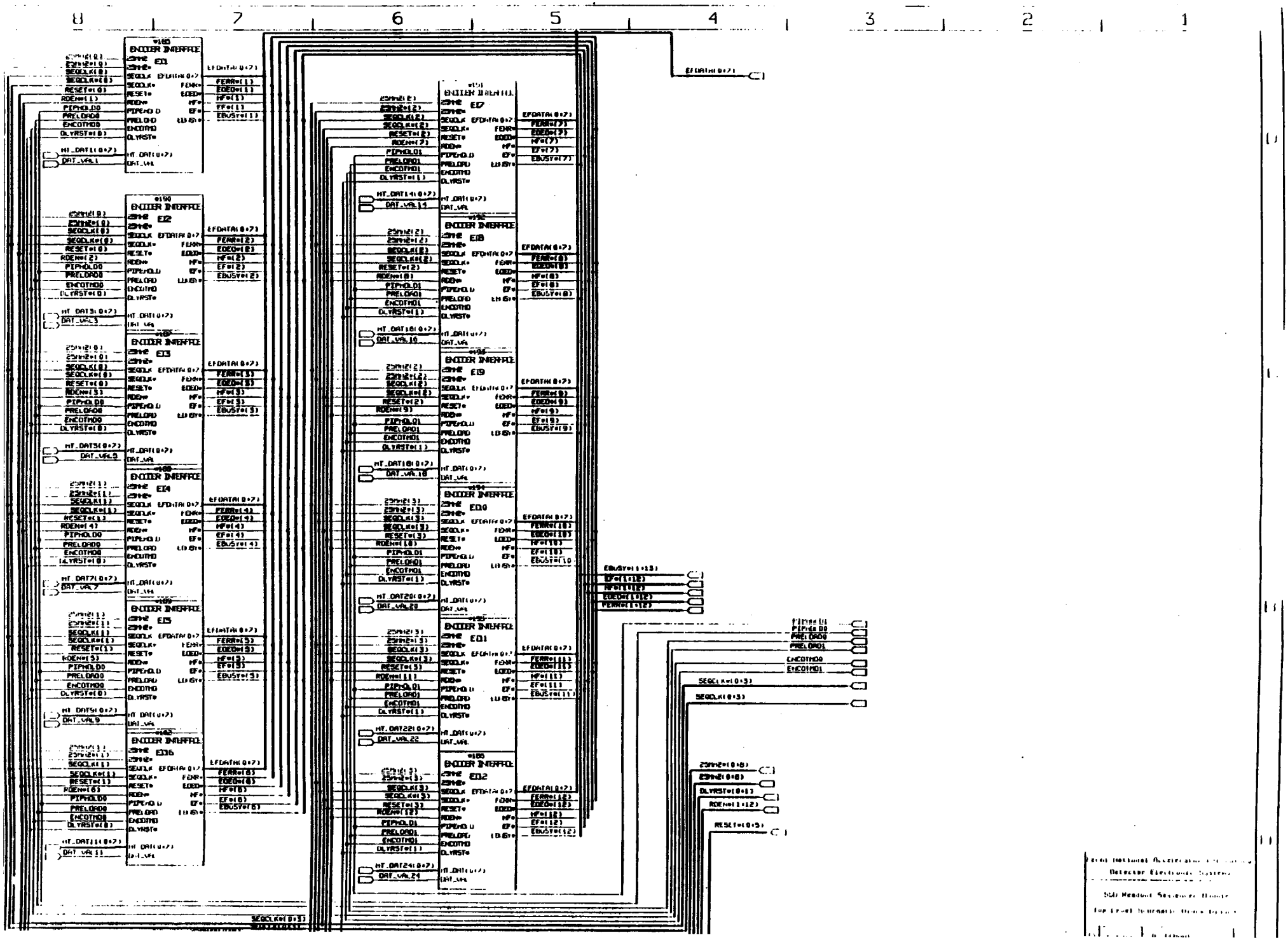
Read the Error Status Register from FASTBUS.

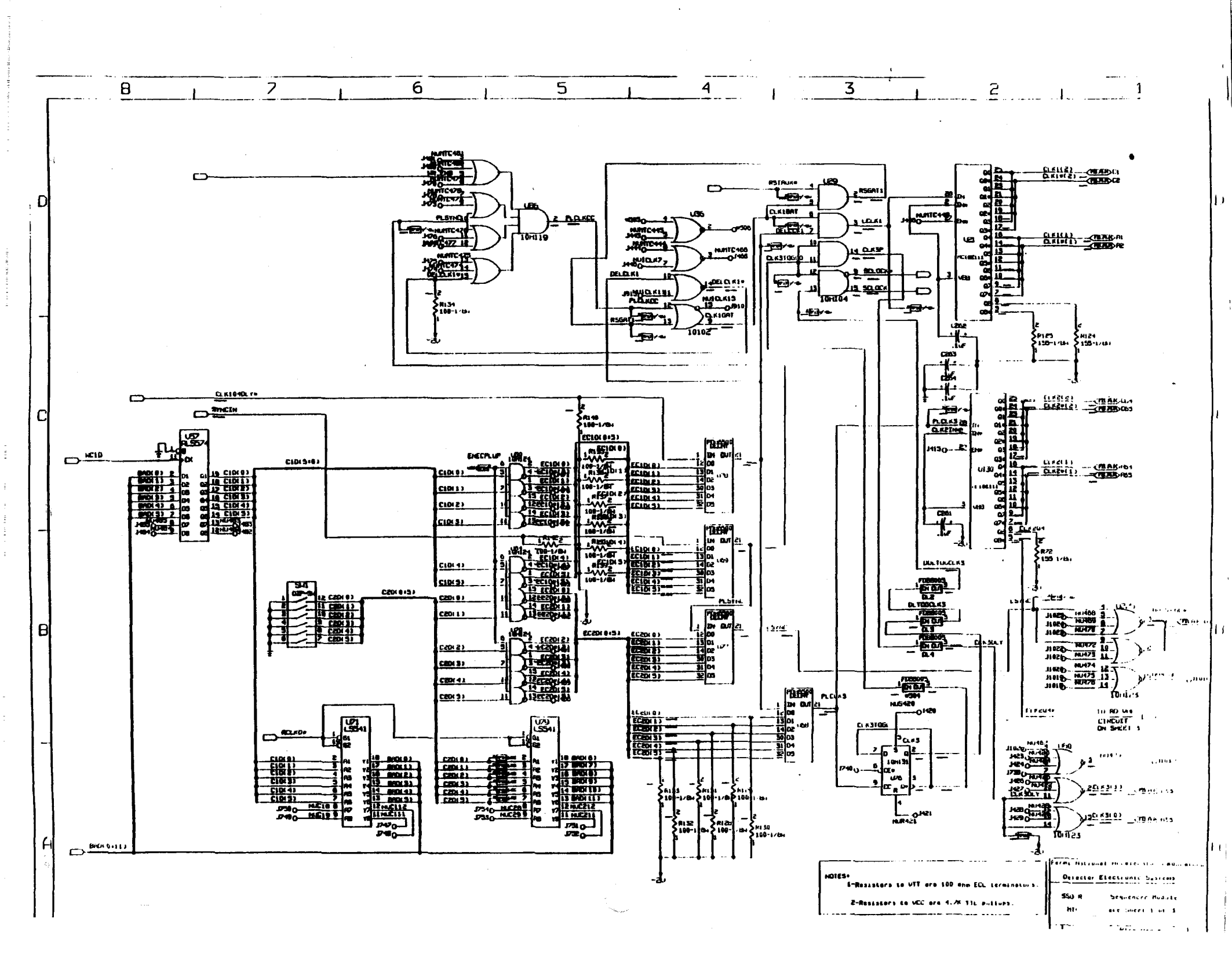
5.2.2. Documented Listing of Each Test Software Module

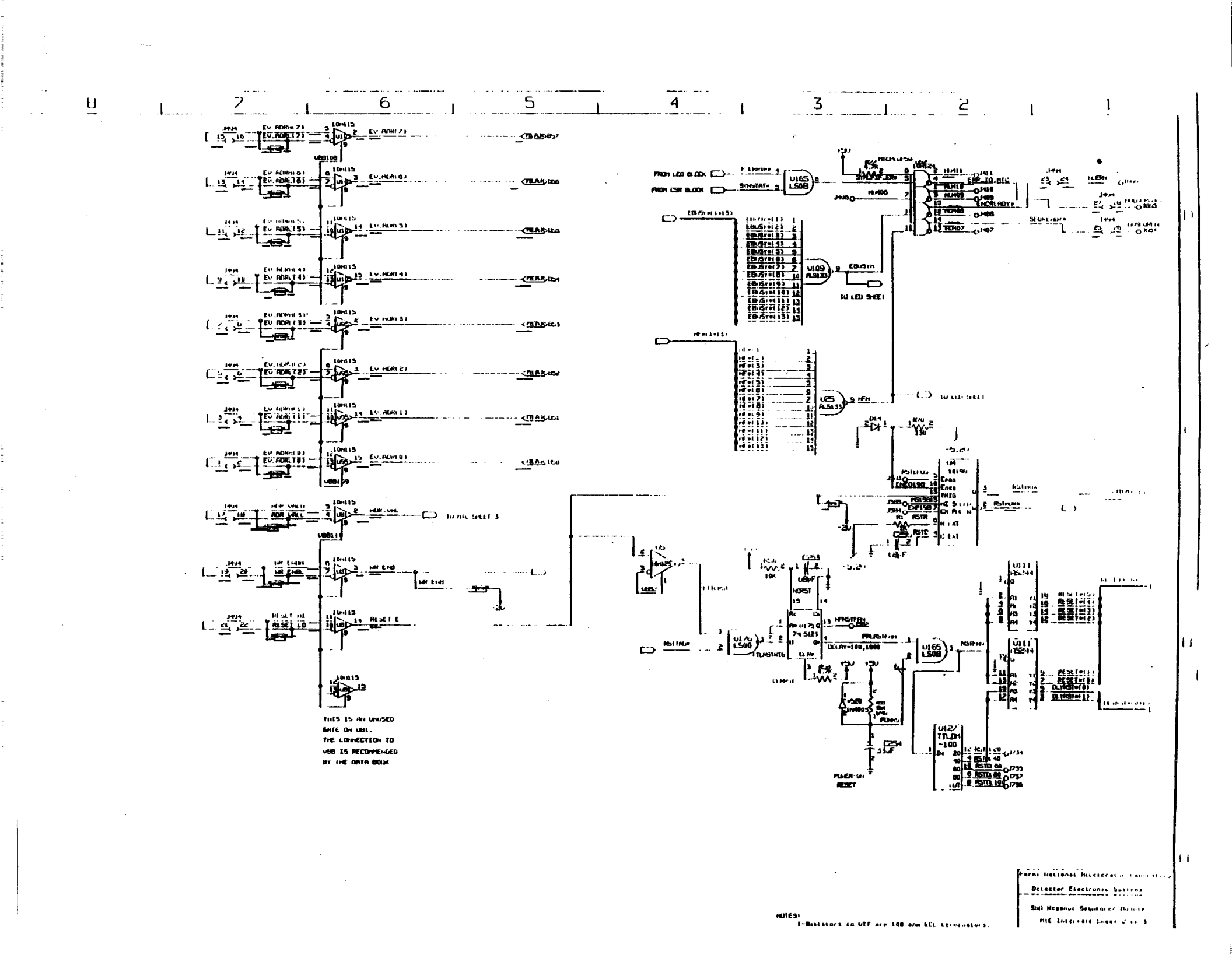
APPENDIX A

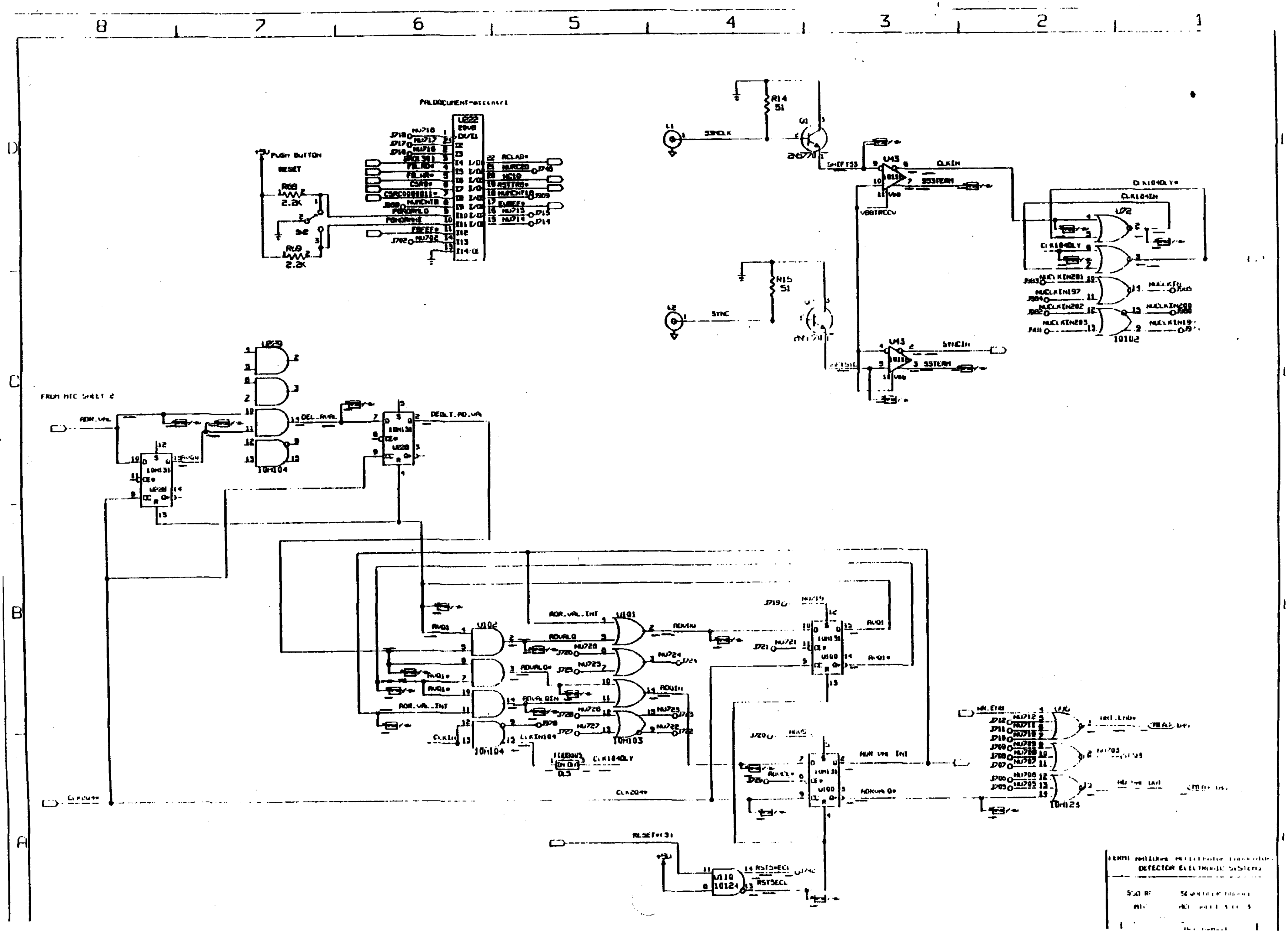
CIRCUIT DIAGRAMS

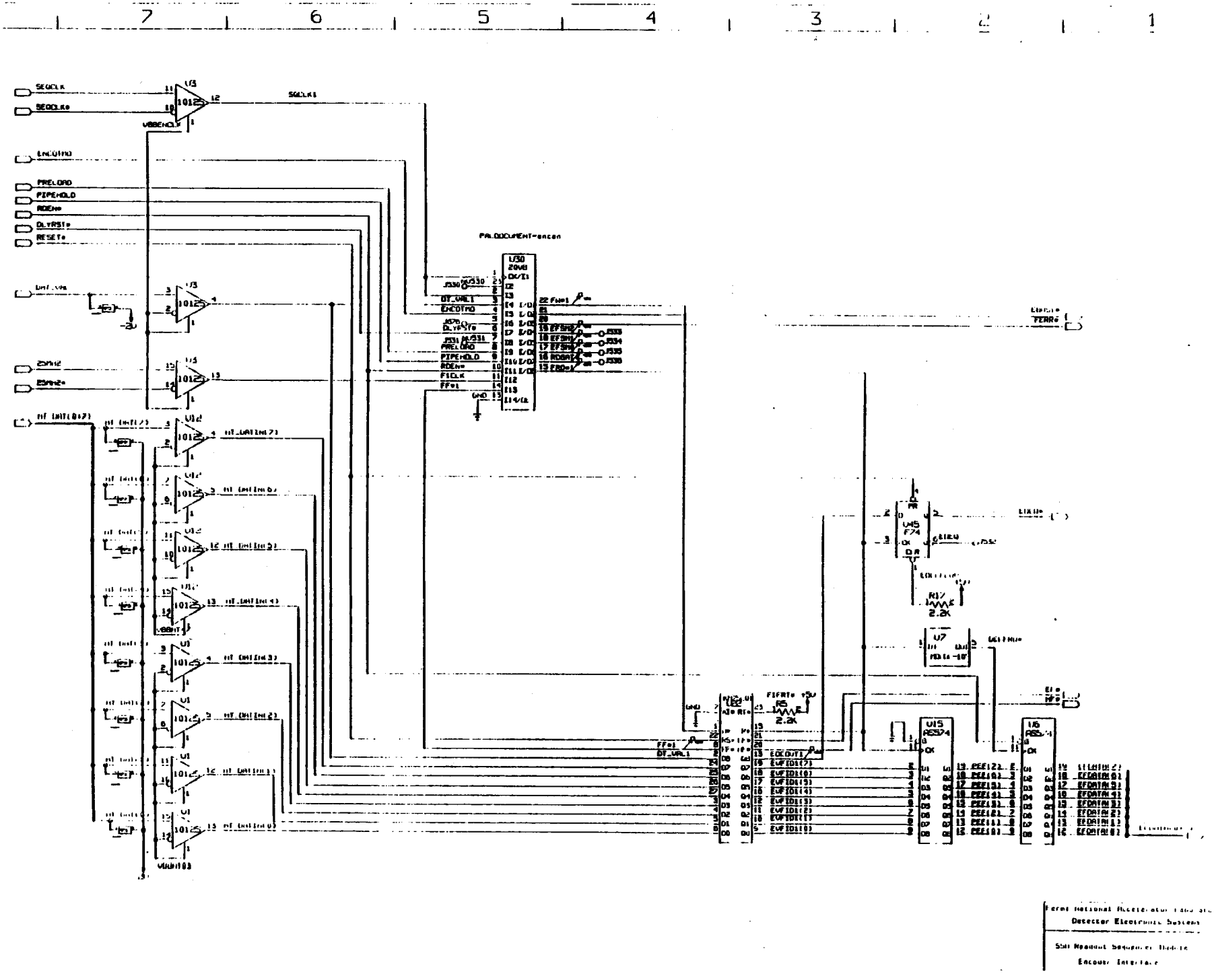

BLOCK/WORD COUNTER
CSR REGISTERS
FASTBUS SLAVE
ARRAY CONTROL
LEDS
INDEX OF DRAWINGS
1 TOP LEVEL SHEET 1
2 TOP LEVEL SHEET 2
3 MTC INTERFACE SHEET 3
4 MTC INTERFACE SHEET 4
5 MTC INTERFACE SHEET 5
6 ENCODER INTERFACE
7 ARRAY CONTROL
8 BLOCK/WORD COUNTER
9 BLOCK/WORD COUNTER SHEET 2
10 FASTBUS EVENT BUFFER
11 AUXILIARY INTERFACE
12 CSR REGISTERS
13 FASTBUS INTERFACE
14 FASTBUS TRANSCEIVER
15 FASTBUS PALS
16 LED DISPLAY
17 INTERFACE CONNECTOR


8
7
6
5
4
3
2
1
ENCODER INTERFACE
EI1
EI2
EI3
EI4
EI5
EI6
EI7
EI8
EI9
EI0
EI1
EI2
EFDATA[0:7]
SEQCLK[0:3]
SEQCLK[4:7]
ENCOTM00
ENCOTM01
PRELOD00
PRELOD01
PIPEHOLD00
PIPEHOLD01
DLYRST[0:3]
DLYRST[4:7]
RDEN[0:3]
RDEN[4:7]
RESET[0:3]
RESET[4:7]
EBUSY[0:3]
EBUSY[4:7]
EBUSY[8:11]
EBUSY[12:15]
EF[0:3]
EF[4:7]
EF[8:11]
EF[12:15]
HF[0:3]
HF[4:7]
HF[8:11]
HF[12:15]
EOED[0:3]
EOED[4:7]
EOED[8:11]
EOED[12:15]
FERR[0:3]
FERR[4:7]
FERR[8:11]
FERR[12:15]
HT_DAT[0:7]
DAT_VAL
Detector Electronics Systems


8
7
6
5
4
3
2
1
D
C
B
A
10H119
10H104
10H102
10H125
NOTES:
1-Resistors to VTT are 100 ohm ECL terminators.
2-Resistors to VCC are 1.2K TTL pullups.
Detector Electronic Systems
SSD R
Sequencer Module

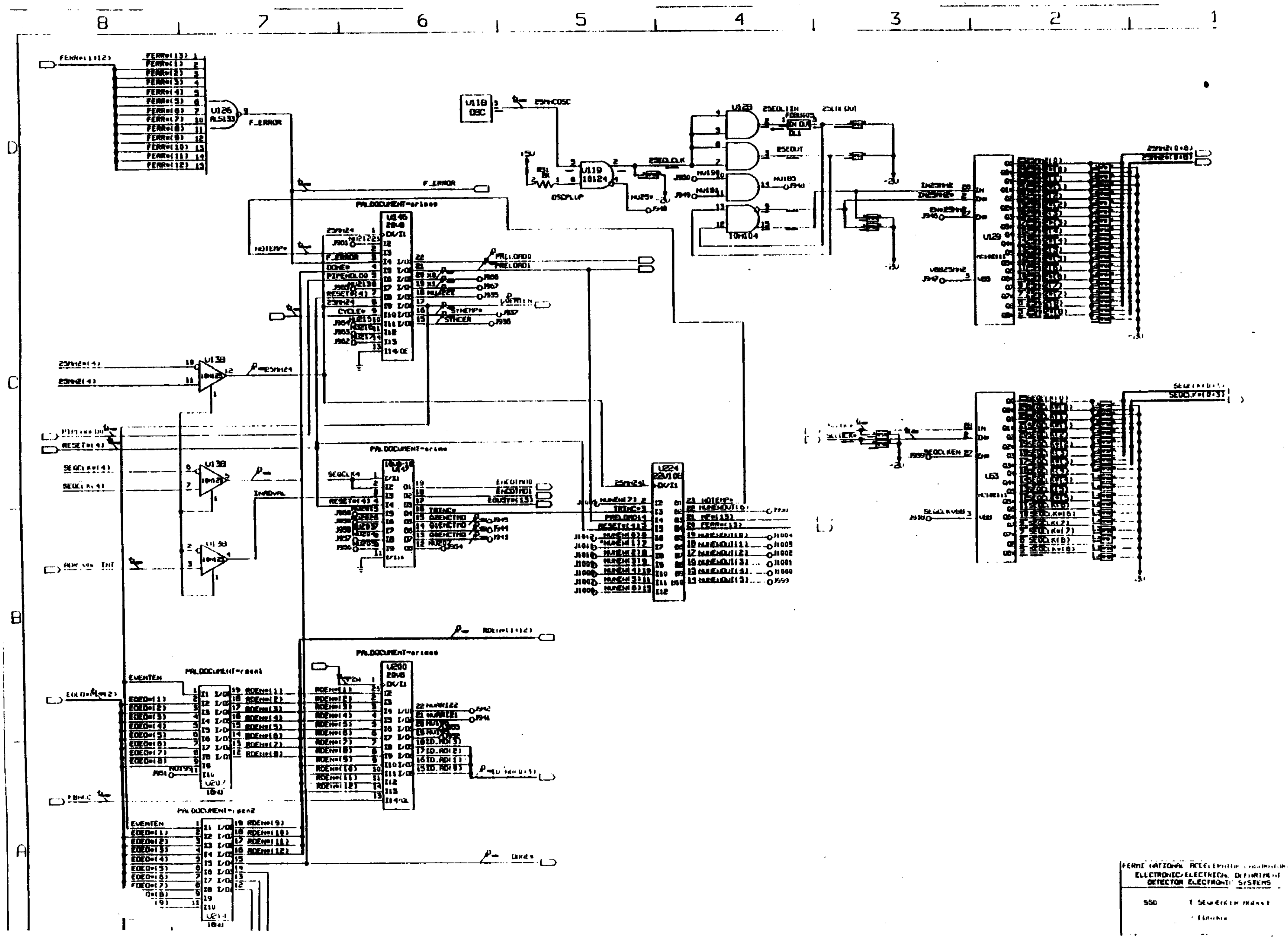

8
7
6
5
4
3
2
1
FROM LED BLOCK
FROM CSR BLOCK
TO LED SHEET
TO MIC SHEET 3
THIS IS AN UNUSED
GATE ON U81.
THE CONNECTION TO
V88 IS RECOMMENDED
BY THE DATA BOOK
NOTES:
1-Resistors in VTF are 100 ohm ECL terminators.
Detector Electronics Systems
RESET


DETECTOR ELECTRONIC SYSTEMS
PUSH BUTTON
RESET
FROM MIC SHEET 2
ADR.VAL
DEL.AVAL
DEGLT.AD.VA
CLKIN
SYNCIN
SSTERM
SSCLK
SYNC
R14 51
R15 51
R68 2.2K
R69 2.2K
U43
U72
U101
U102
U110
U222
10H104
10H103
10H125
10102
10124
10H131
ADR.VAL.INT
ADVALO
ADVALQ
ADVIN
ADVAL QIN
CLK20
RESET
RSTSECL

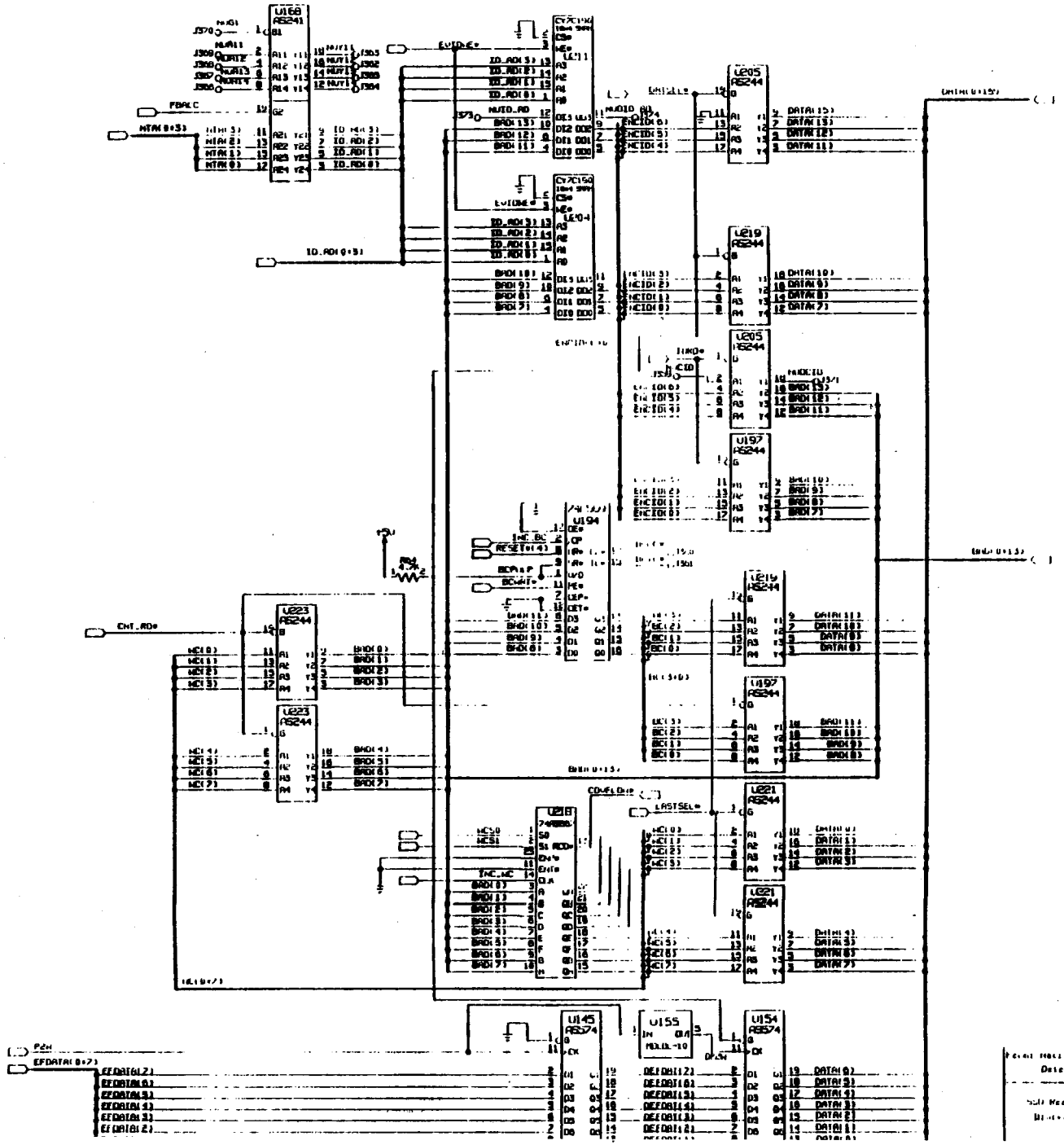


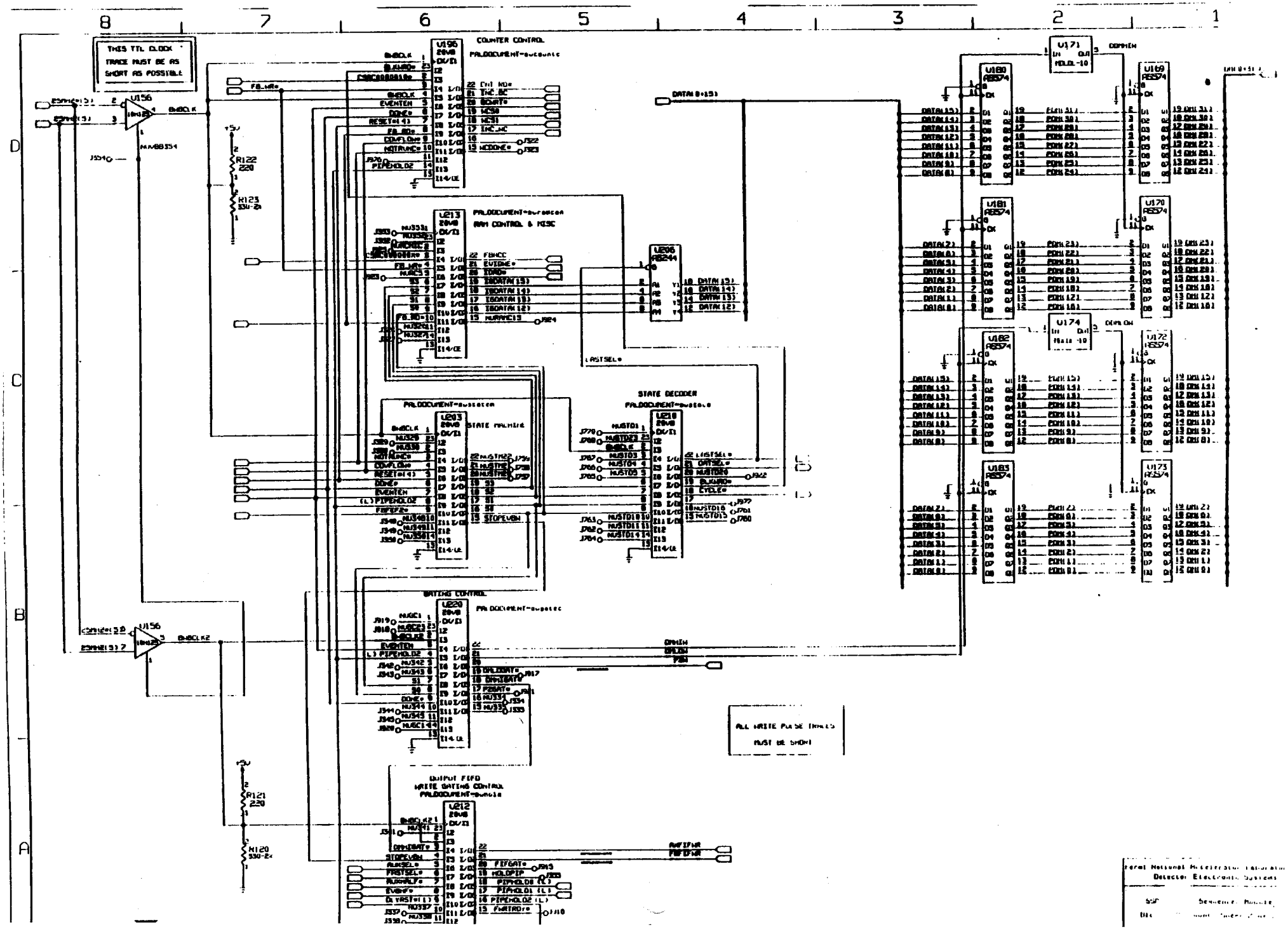
Fermi National Accelerator Laboratory
Detector Electronic Systems

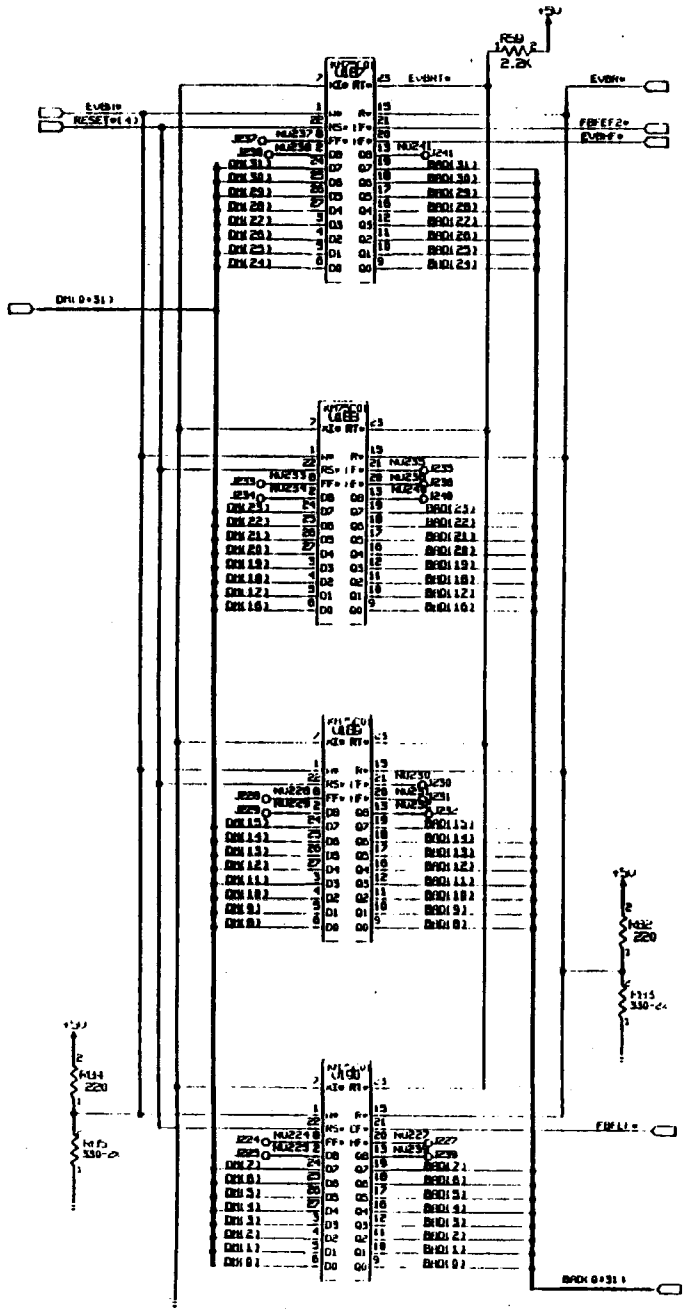
Star Readout Sequencer Module
Encoder Interface



FERMI NATIONAL ACCELERATOR LABORATORY
ELECTRONIC/ELECTRICAL DEPARTMENT
DETECTOR ELECTRONIC SYSTEMS
550
U118 OSC
25MHCOSC
U119 10124
OSCPLUP
U128
10H104
U126 ALS133
F_ERROR
U129
U63
U224 22V10B
PALDOCUMENT=rxeni
PALDOCUMENT=rxen2
U138
U200
U146
PRELOAD0
PRELOAD1
SYNCER
INADVAL
RESET*(4)
EVENTEN



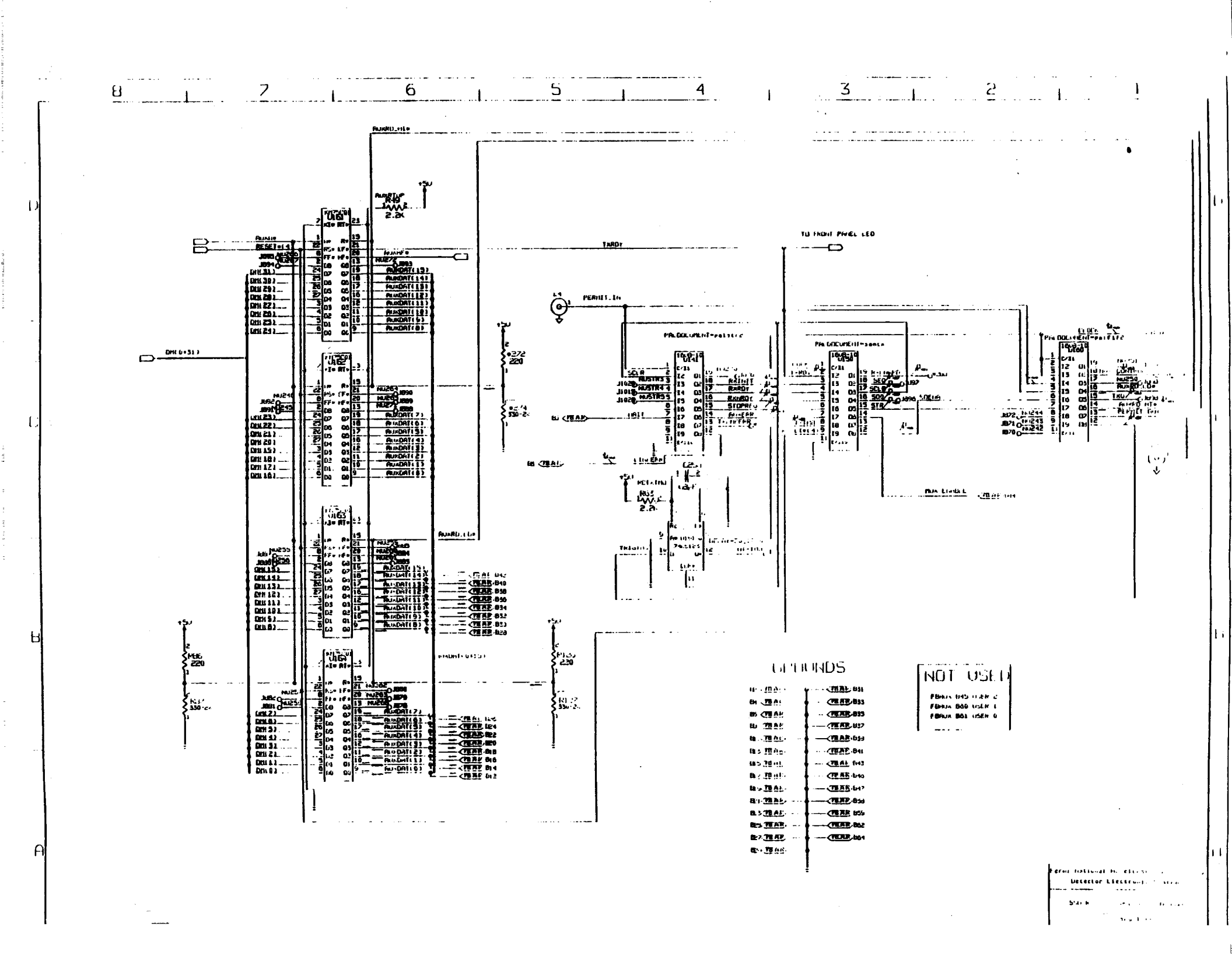

THIS TTL CLOCK TRACE MUST BE AS SHORT AS POSSIBLE
COUNTER CONTROL
RAM CONTROL & MISC
STATE MACHINE
STATE DECODER
BYTING CONTROL
Output FIFO WRITE BYTING CONTROL
ALL WRITE PULSE TIMES MUST BE SHORT
Federal National Mortgage Association
Document Electronic Systems

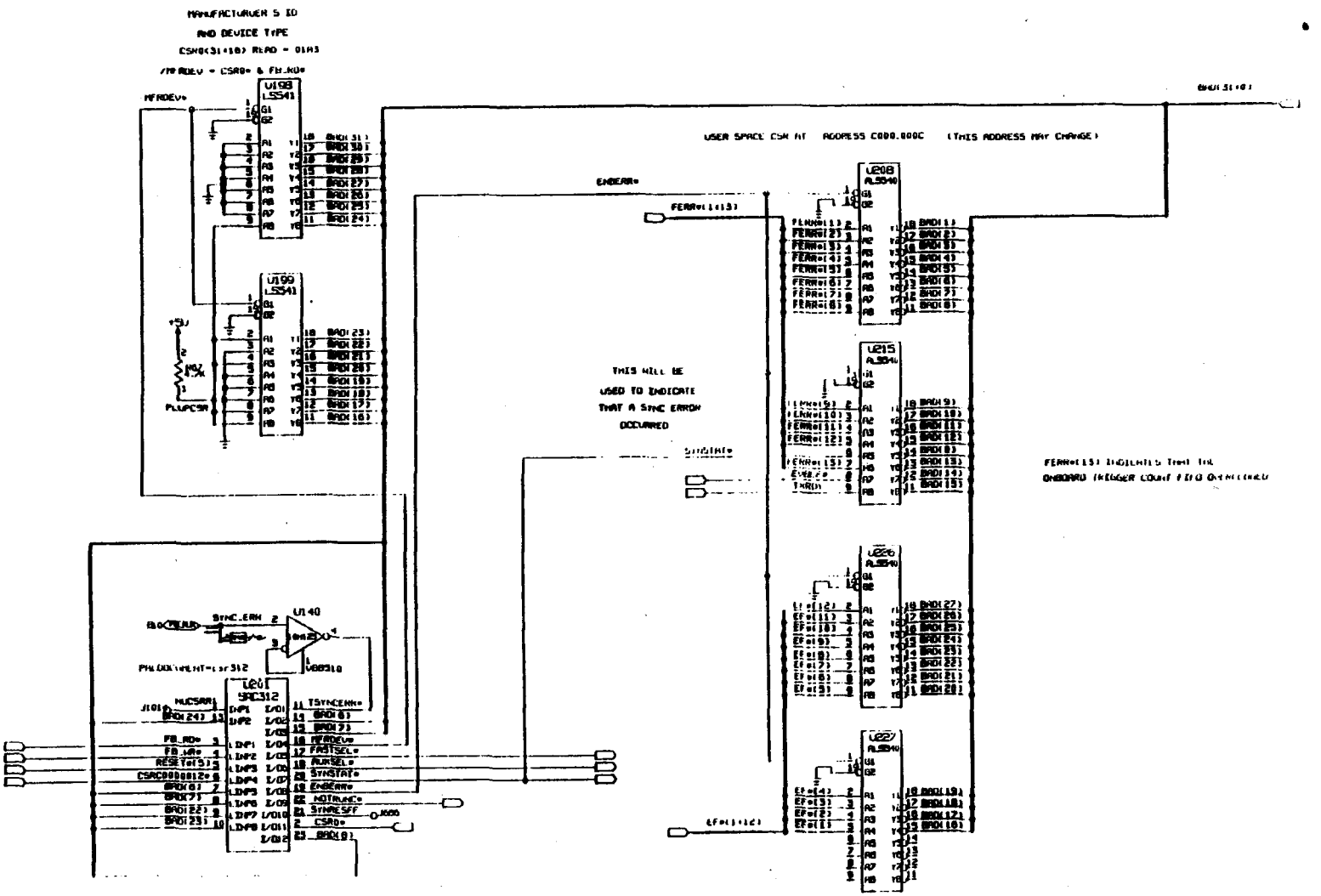


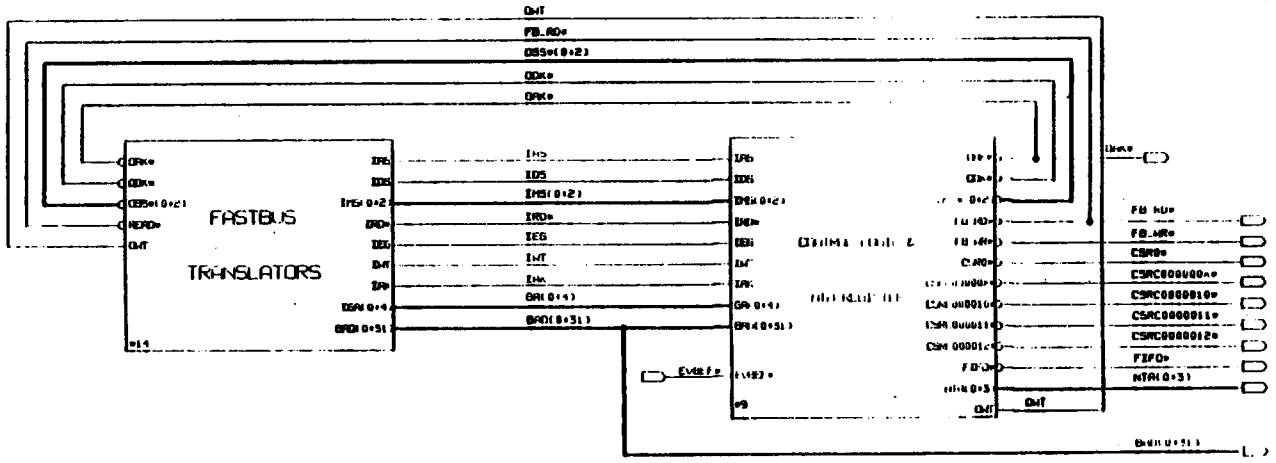
Fermi National Accelerator Laboratory
Detector Electronic Systems

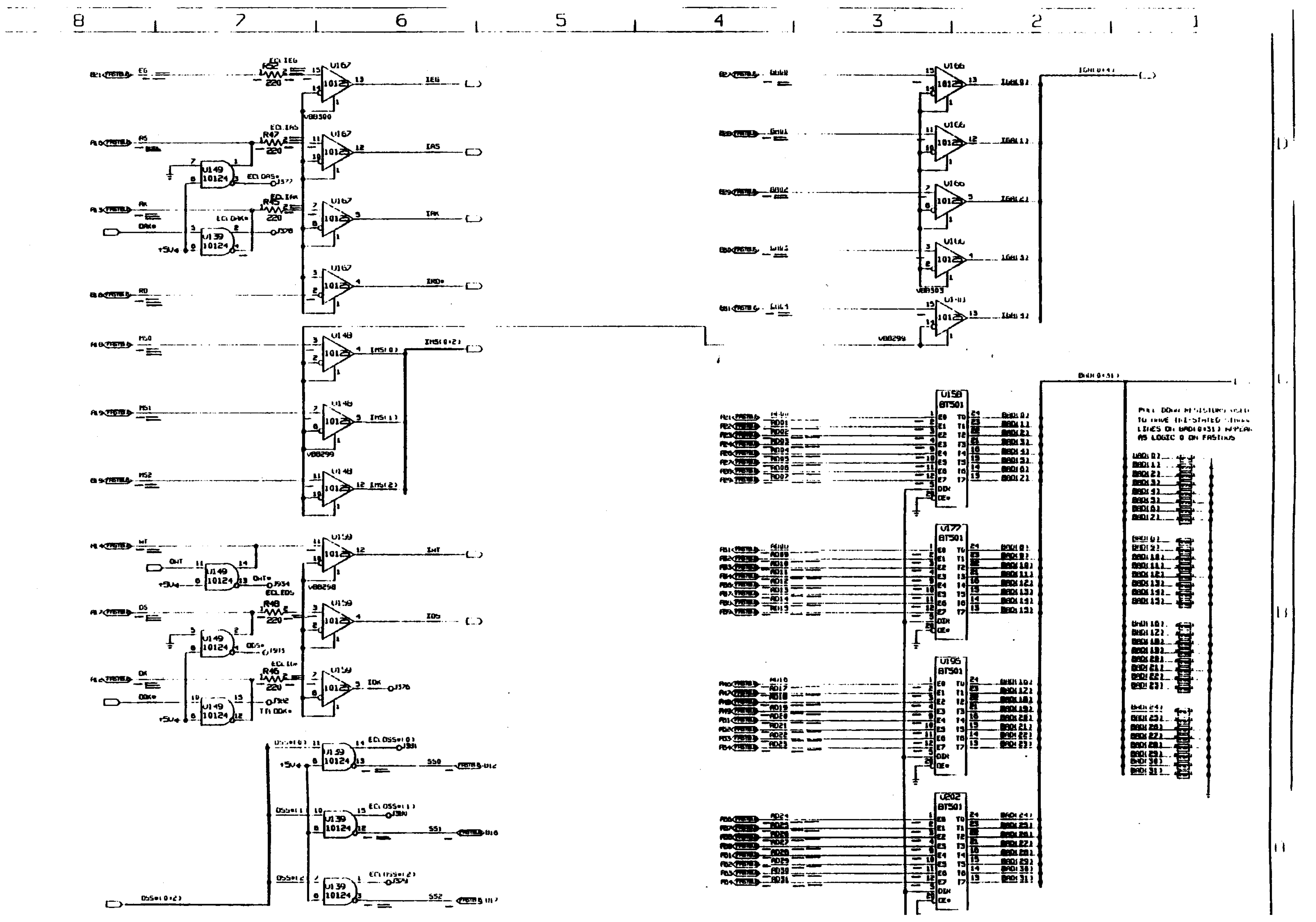
SVX Readout Sequencer Module

Sheet Title

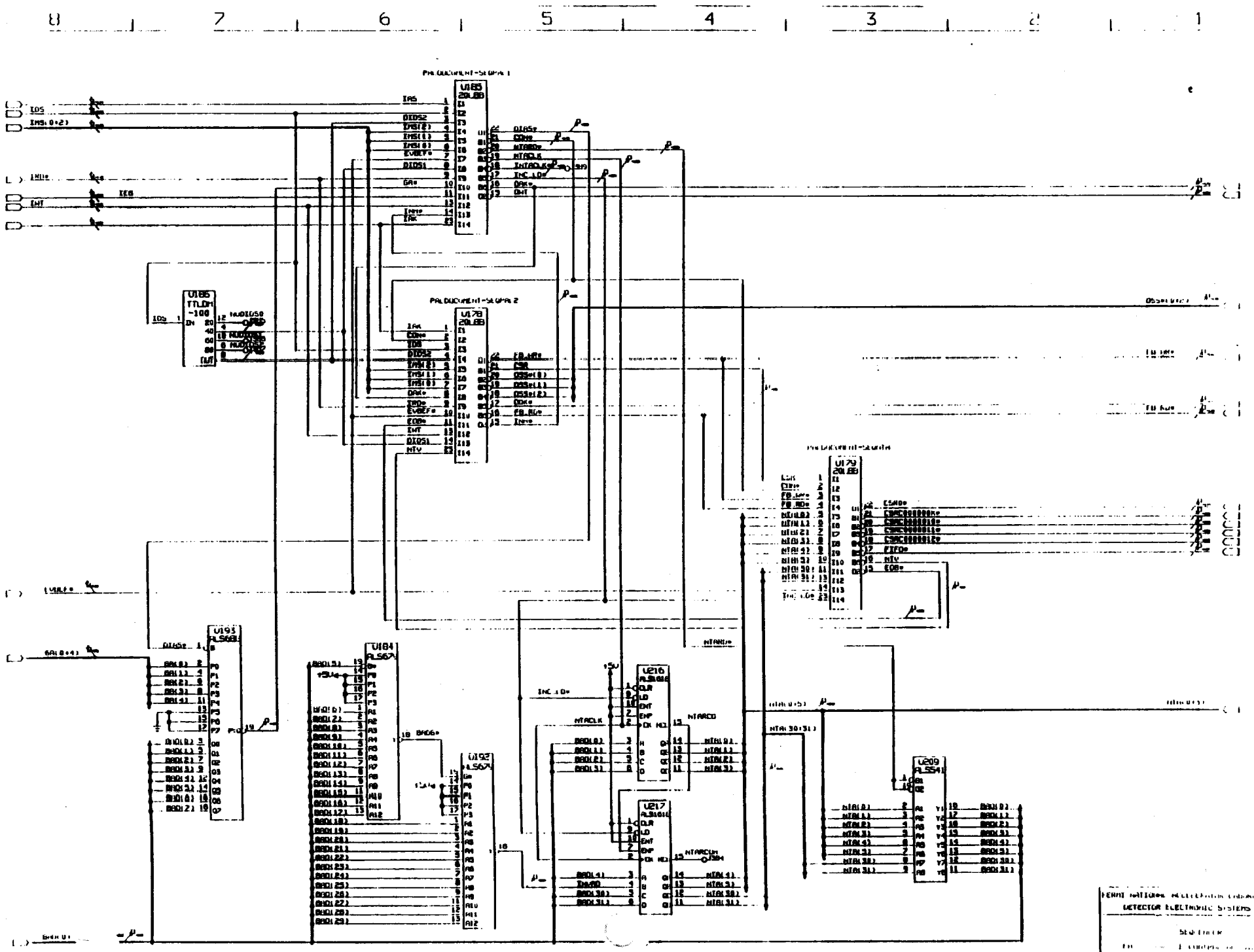

8
7
6
5
4
3
2
1
D
C
B
A
TO FRONT PANEL LED
GROUNDS
NOT USED

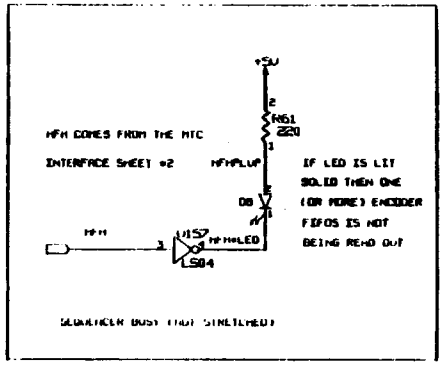

MANUFACTURER'S ID
AND DEVICE TYPE
CSR0(31:16) READ = 0183
/MFRDEV = CSR0* & FB_RD*
MFRDEV*
U198
LS541
U199
LS541
+5V
4K
PULLUPCSR
USER SPACE CSR AT ADDRESS C000.000C (THIS ADDRESS MAY CHANGE)
ENBERR*
FERR*(15:0)
U208
AL5040
U215
AL5040
U226
AL5040
U227
AL5040
THIS WILL BE
USED TO INDICATE
THAT A SYNC ERROR
OCCURRED
SYNCERR*
U140
SYNC_ERR
SRC312
NUCSR*
FB_RD*
RESET*
CSRC000C12
TSYNCERR*
MFRDEV*
FASTSEL*
MUXSEL*
SYNSTAT*
ENBERR*
MOTRUN*
SYNCSFF
CSR0*
FERR*(15:0) INDICATES THAT THE
ONBOARD TRIGGER COUNT FIFO OVERFLOWED
FERMI NATIONAL ACCELERATOR LABORATORY
ELECTRONICS/ELECTRICAL DEPARTMENT
DETECTOR ELECTRONIC SYSTEMS
SSD READOUT SEQUENCER MODULE
CONTROL AND STATUS REGISTERS
D
1/19/90

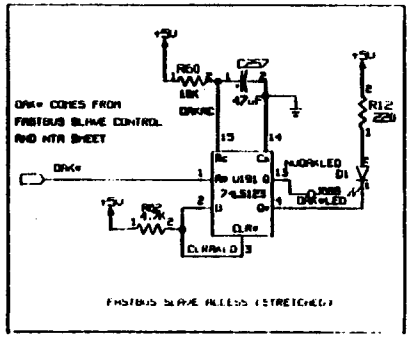

FASTBUS
TRANSLATORS

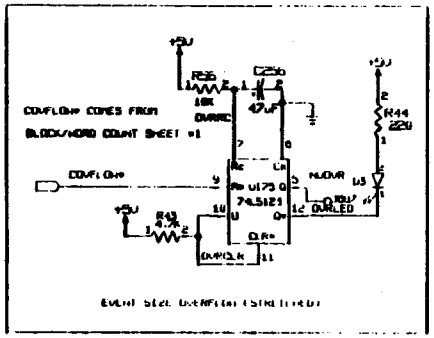


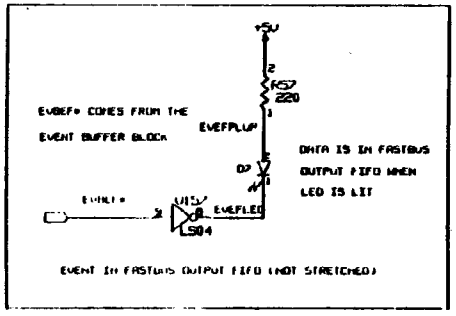
8
7
6
5
4
3
2
1
U167
10124
U166
U148
U149
U139
U159
U158
BT501
U177
U195
U202
R45
R46
R47
R48
R52
220
+5V4
IEG
IAS
IAK
IRD*
IMS(0:2)
IWT
IDS
IDK
SS0
SS1
SS2
DSS*(0:2)
BAD(0:31)
PULL DOWN RESISTORS USED TO HAVE TRI-STATED LINES ON BAD(0:31) APPEAR AS LOGIC 0 ON FASTBUS

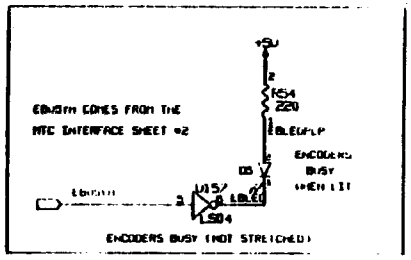

8
7
6
5
4
3
2
1
U185
20L8B
U178
20L8B
U179
20L8B
U191
RLS688
U184
RLS674
U192
RLS674
U216
RLS818
U217
RLS818
U209
RLS541
U185
TTLDH
-100
PAL DOCUMENT-SEGMENT 1
PAL DOCUMENT-SEGMENT 2
PAL DOCUMENT-SEGMENT 3
DETECTOR ELECTRONIC SYSTEMS

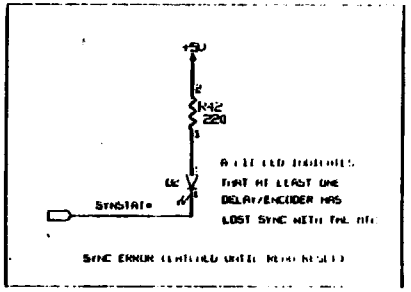


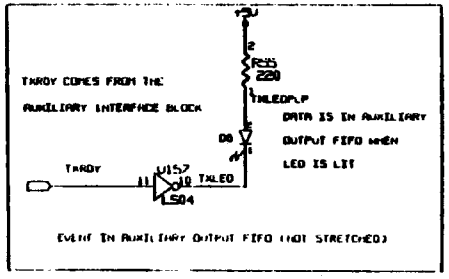


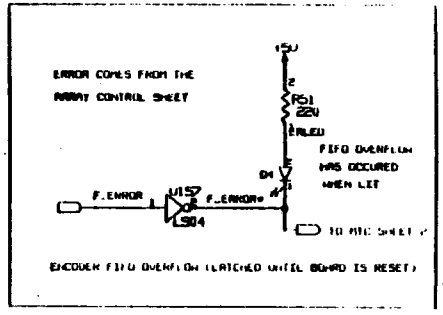


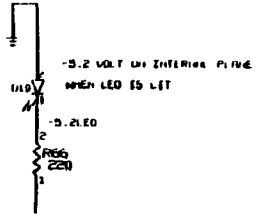


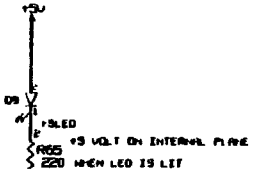






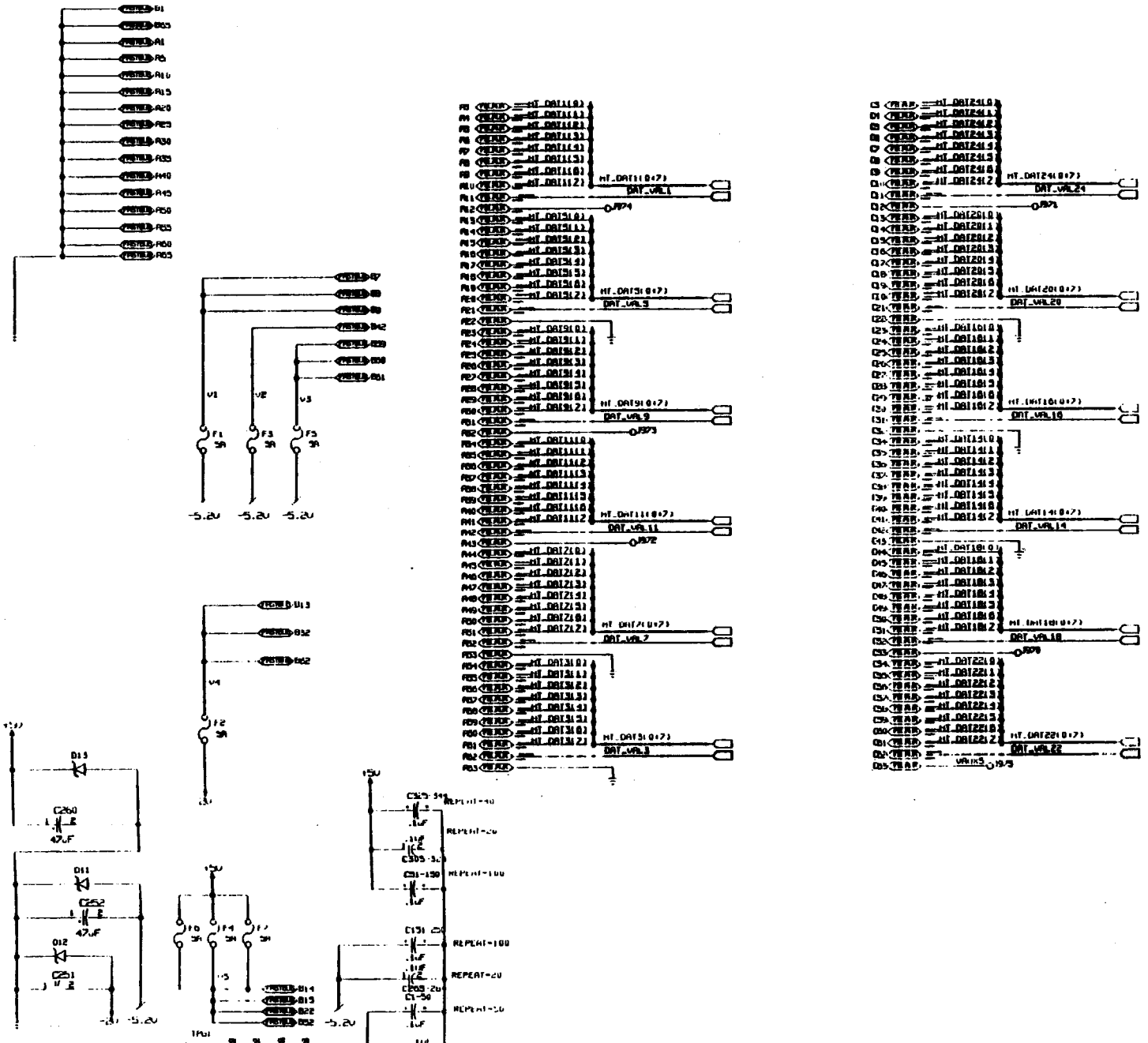






FERMI NATIONAL ACCELERATOR LABORATORY
ELECTRONICS/ELECTRICAL DEPARTMENT
DETECTOR ELECTRONICS SYSTEMS

SSO READOUT SEQUENCER MODULE
LED DRIVER CIRCUITS


8
7
6
5
4
3
2
1
-5.2V
+5V
DAT_VAL1
DAT_VAL9
DAT_VAL11
DAT_VAL7
DAT_VAL3
DAT_VAL24
DAT_VAL20
DAT_VAL16
DAT_VAL14
DAT_VAL18
DAT_VAL22
REPEAT=100
REPEAT=20
REPEAT=10

APPENDIX B

PROGRAMMABLE LOGIC EQUATIONS

ARIDAD.abl ID ADDRESS ENCODER

```
MODULE ID_ADDRESS_ENCODER
FLAG '-R7'
TITLE 'ANDY ROMERO RD/EE CONTROLS 2-6-90'

ARIDAD DEVICE 'P20V8R';

CLK,RDEN1,RDEN2,RDEN3,RDEN4,RDEN5,RDEN6    PIN  1,23,2,3,4,5,6;
RDEN7,RDEN8,RDEN9,RDEN10,RDEN11,RDEN12     PIN  7,8,9,10,11,14;
IDAD3,IDAD2,IDAD1,IDAD0                    PIN  18,17,16,15;
FBACC                                      PIN  13;

"Standard Symbol Declarations"
X=.X.;
C=.C.;
P=.P.;
Z=.Z.;
"End Standard Symbol Declarations"

VECIN =
[RDEN1,RDEN2,RDEN3,RDEN4,RDEN5,RDEN6,RDEN7,RDEN8,RDEN9,RDEN10,RDEN11,RDEN12];
ID_AD_VEC = [IDAD3,IDAD2,IDAD1,IDAD0];

TRUTH_TABLE ([VECIN] :> [ID_AD_VEC])

[[0,1,1,1,1,1,1,1,1,1,1,1]] :> [[0,0,0,0]];
[[1,0,1,1,1,1,1,1,1,1,1,1]] :> [[0,0,0,1]];
[[1,1,0,1,1,1,1,1,1,1,1,1]] :> [[0,0,1,0]];
[[1,1,1,0,1,1,1,1,1,1,1,1]] :> [[0,0,1,1]];
[[1,1,1,1,0,1,1,1,1,1,1,1]] :> [[0,1,0,0]];
[[1,1,1,1,1,0,1,1,1,1,1,1]] :> [[0,1,0,1]];
[[1,1,1,1,1,1,0,1,1,1,1,1]] :> [[0,1,1,0]];
[[1,1,1,1,1,1,1,0,1,1,1,1]] :> [[0,1,1,1]];
[[1,1,1,1,1,1,1,1,0,1,1,1]] :> [[1,0,0,0]];
[[1,1,1,1,1,1,1,1,1,0,1,1]] :> [[1,0,0,1]];
[[1,1,1,1,1,1,1,1,1,1,0,1]] :> [[1,0,1,0]];
[[1,1,1,1,1,1,1,1,1,1,1,0]] :> [[1,0,1,1]];
[[1,1,1,1,1,1,1,1,1,1,1,1]] :> [[1,1,1,1]];

TEST_VECTORS

([CLK,FBACC,VECIN] -> [ID_AD_VEC])

"[C,1,[X,X,X,X,X,X,X,X,X,X,X,X]] -> [[Z,Z,Z,Z]]; Zs cause unisite to barf"
[C,0,[0,1,1,1,1,1,1,1,1,1,1,1]] -> [[0,0,0,0]];
[C,0,[1,0,1,1,1,1,1,1,1,1,1,1]] -> [[0,0,0,1]];
[C,0,[1,1,0,1,1,1,1,1,1,1,1,1]] -> [[0,0,1,0]];
[C,0,[1,1,1,0,1,1,1,1,1,1,1,1]] -> [[0,0,1,1]];
[C,0,[1,1,1,1,0,1,1,1,1,1,1,1]] -> [[0,1,0,0]];
[C,0,[1,1,1,1,1,0,1,1,1,1,1,1]] -> [[0,1,0,1]];
[C,0,[1,1,1,1,1,1,0,1,1,1,1,1]] -> [[0,1,1,0]];
[C,0,[1,1,1,1,1,1,1,0,1,1,1,1]] -> [[0,1,1,1]];
[C,0,[1,1,1,1,1,1,1,1,0,1,1,1]] -> [[1,0,0,0]];
[C,0,[1,1,1,1,1,1,1,1,1,0,1,1]] -> [[1,0,0,1]];
[C,0,[1,1,1,1,1,1,1,1,1,1,0,1]] -> [[1,0,1,0]];
[C,0,[1,1,1,1,1,1,1,1,1,1,1,0]] -> [[1,0,1,1]];
[C,0,[1,1,1,1,1,1,1,1,1,1,1,1]] -> [[1,1,1,1]];

END ID_ADDRESS_ENCODER
```

```
[C, X,X, 1,1,X,1, [X,X]]->[[0,1]];
[C, X,X, X,X,X,X, [X,X]]->[[1,1]];
[C, X,X, X,X,1,1, [X,X]]->[[1,1]];
[C, X,X, X,X,1,0, [X,X]]->[[0,0]];

END FACU_CONTROL
```

ARTMO.abl GLOBAL TIMEOUT CHIP

```
MODULE GLOBAL_TIMEOUT_CHIP
FLAG '-R7'
TITLE 'ANDY ROMERO RD/EE CONTROLS 2-2-90'

"6-4-90 Because the use of the encoder fifo empty flags for
"       encoder fifo readout initiation was a stupid idea,
"       we will have to modify this chip as part a new encoder
"       fifo readout cycle initiation scheme. See the design note
"       for details on the operation of this new scheme.
"       -Add the signal TRINC (trigger count increment) to pin 16
"        -Move Q2,Q1,Q0 from pins 16,15,14 to pins 15,14,13
"       -TRINC will be the MSB of the state vector


ARTMO DEVICE 'P16V8R';

PIN1,PIN2,PIN3,PIN4    PIN  1,2,3,4;
PIN19,PIN18,PIN17,PIN16,PIN15,PIN14,PIN13  PIN  19,18,17,16,15,14,13;

"Declaration of active levels"

SEQCLK  = PIN1;
SEQCLKB = PIN2;
ADR_VAL  = PIN3;
RESET  = !PIN4;

ENCOTMO0 = !PIN19;
ENCOTMO1 = !PIN18;
EBUSY   = !PIN17;
TRINC   = PIN16;
Q2      = PIN15;
Q1      = PIN14;
Q0      = PIN13;


"Standard Symbol Declarations"

X=.X.;
C=.C.;
P=.P.;



STATEVEC = [TRINC,Q2,Q1,Q0];

TRINC_HOLD = [0,0,0,0];
TRAP1 = [0,0,0,1];
TRAP2 = [0,0,1,0];
TRAP3 = [0,0,1,1];
TRAP4 = [0,1,0,0];
TRAP5 = [0,1,0,1];
TRAP6 = [0,1,1,0];
TIMEOUT = [0,1,1,1];
WAIT = [1,0,0,0];
S1 = [1,0,0,1];
S2 = [1,0,1,0];
S3 = [1,0,1,1];
S4 = [1,1,0,0];
```

ARLOAD.abl ARRAY CONTROL UNIT STATE MACHINE

```
MODULE FACU_CONTROL
FLAG '-R7'
TITLE 'ANDY ROMERO RD/EE CONTROLS 2-8-90'

"This Chip needs the special reduction technique"

ARLOAD DEVICE 'P20V8R';

PIN1,PIN2,PIN3,PIN4,PIN5,PIN7,PIN8,PIN9  PIN  1,2,3,4,5,7,8,9;
PIN15,PIN16,PIN17,PIN19,PIN20,PIN21,PIN22  PIN 15,16,17,19,20,21,22;

"Declaration of Active Levels"

CLK = PIN1;          PRELOAD0 = !PIN22;
NOTEMP = !PIN2;      PRELOAD1 = !PIN21;
ERROR = PIN3;        X0 = PIN20;
DONE = !PIN4;        X1 = PIN19;
PIPEHOLD = !PIN5;    EVENTEN = !PIN17;
RESET = !PIN7;       SYNCD_NOTEMP = !PIN16;
CLKB = PIN8;         SYNCD_ERROR = PIN15;
CYCLE = !PIN9;

STATEVEC = [X1,X0];

HOLD = [0,0];
FULL_ERROR = [1,0];
PRE_LOAD = [0,1];
EVENT_READOUT = [1,1];


"Standard Symbol Declarations"

X=.X.;
C=.C.;
P=.P.;
Z=.Z.;

"End Standard Symbol Declarations"

EQUATIONS

PRELOAD0 = (
        ((STATEVEC == PRE_LOAD) & !CLKB)
       #(PRELOAD0 & (STATEVEC == PRE_LOAD))
        #(PRELOAD0 & CLKB)
       );

PRELOAD1 = (
        ((STATEVEC == PRE_LOAD) & !CLKB)
       #(PRELOAD1 & (STATEVEC == PRE_LOAD))
        #(PRELOAD1 & CLKB)
       );

EVENTEN = (
       ((STATEVEC == EVENT_READOUT) & !CLKB)
      #(EVENTEN & (STATEVEC == EVENT_READOUT))
       #(EVENTEN & CLKB)
      );
```

```
SYNCD_NOTEMP := NOTEMP;

SYNCD_ERROR := ERROR;


STATE_DIAGRAM STATEVEC

STATE HOLD:;

IF RESET THEN HOLD
ELSE
  IF SYNCD_ERROR THEN FULL_ERROR
  ELSE
    IF (SYNCD_NOTEMP & !PIPEHOLD & !CYCLE) THEN PRE_LOAD
    ELSE HOLD;

STATE FULL_ERROR:;

IF RESET THEN HOLD
ELSE FULL_ERROR;

STATE PRE_LOAD:;

GOTO EVENT_READOUT;

STATE EVENT_READOUT:;

IF (RESET # (DONE & !PIPEHOLD)) THEN HOLD
ELSE EVENT_READOUT;


TEST_VECTORS

"([CLK, ERROR,NOTEMP, PIPEHOLD,CYCLE,DONE,RESET, STATEVEC]->[STATEVEC])"

([PIN1, PIN3,PIN2, PIN5,PIN9,PIN4,PIN7, STATEVEC]->[STATEVEC])

[P, 0,1, 1,1,1,1, [0,0]]->[[X,X]];
[C, 0,1, 1,1,1,0, [X,X]]->[[0,0]];
[P, 0,1, 1,1,1,1, [0,0]]->[[X,X]];
[C, 0,1, 1,1,1,0, [X,X]]->[[0,0]];
[C, 1,1, 1,1,1,1, [X,X]]->[[0,0]];
[C, X,X, X,X,X,1, [X,X]]->[[1,0]];
[P, 0,1, 1,1,1,1, [0,0]]->[[X,X]];
[C, 0,1, 1,1,1,0, [X,X]]->[[0,0]];
[C, 0,0, X,X,X,X, [X,X]]->[[0,0]];
[C, X,X, 1,1,X,1, [X,X]]->[[0,1]];

[P, 0,1, 1,1,1,1, [0,0]]->[[X,X]];
[C, 0,1, 1,1,1,0, [X,X]]->[[0,0]];
[C, 0,0, X,X,X,1, [X,X]]->[[0,0]];
[C, X,X, 1,1,X,1, [X,X]]->[[0,1]];
[C, X,X, X,X,X,X, [X,X]]->[[1,1]];
[C, X,X, X,X,1,1, [X,X]]->[[1,1]];
[C, X,X, 0,X,0,1, [X,X]]->[[1,1]];
[C, X,X, 1,X,0,1, [X,X]]->[[0,0]];

[P, 0,1, 1,1,1,1, [0,0]]->[[X,X]];
[C, 0,1, 1,1,1,0, [X,X]]->[[0,0]];
[C, 0,0, X,X,X,1, [X,X]]->[[0,0]];
```

```
S5 = [1,1,0,1];
S6 = [1,1,1,0];
TRAP7 = [1,1,1,1];

EQUATIONS

EBUSY = (STATEVEC != WAIT);

ENCOTMO0 = (
        (!SEQCLKB & (STATEVEC == TIMEOUT))
       #(ENCOTMO0 & (STATEVEC == TIMEOUT))
        #(ENCOTMO0 & SEQCLKB)
       );

ENCOTMO1 = (
        (!SEQCLKB & (STATEVEC == TIMEOUT))
       #(ENCOTMO1 & (STATEVEC == TIMEOUT))
        #(ENCOTMO1 & SEQCLKB)
       );


STATE_DIAGRAM STATEVEC

STATE WAIT:;

IF RESET THEN WAIT
ELSE
IF ADR_VAL THEN S1
ELSE WAIT;

STATE S1:;

IF RESET THEN WAIT
ELSE S2;

STATE S2:;

IF RESET THEN WAIT
ELSE S3;

STATE S3:;

IF RESET THEN WAIT
ELSE S4;

STATE S4:;

IF RESET THEN WAIT
ELSE S5;

STATE S5:;

IF RESET THEN WAIT
ELSE S6;

STATE S6:;

IF RESET THEN WAIT
ELSE TIMEOUT;

STATE TIMEOUT:;
```

```
IF RESET THEN WAIT
ELSE TRINC_HOLD;


STATE TRINC_HOLD::;

GOTO WAIT;


STATE TRAP1::;
GOTO WAIT;

STATE TRAP2::;
GOTO WAIT;

STATE TRAP3::;
GOTO WAIT;

STATE TRAP4::;
GOTO WAIT;

STATE TRAP5::;
GOTO WAIT;

STATE TRAP6::;
GOTO WAIT;

STATE TRAP7::;
GOTO WAIT;

TEST_VECTORS

([PIN1,PIN3,PIN4,      PIN16,PIN15,PIN14,PIN13]->[PIN16,PIN15,PIN14,PIN13])


[P,0,0,   1,0,0,0]->[X,X,X,X];
[C,0,0,   1,0,0,0]->[X,X,X,X];
[C,1,1,   X,X,X,X]->[1,0,0,1];
[C,0,1,   X,X,X,X]->[1,0,1,0];
[C,0,1,   X,X,X,X]->[1,0,1,1];
[C,0,1,   X,X,X,X]->[1,1,0,0];
[C,0,1,   X,X,X,X]->[1,1,0,1];
[C,0,1,   X,X,X,X]->[1,1,1,0];
[C,0,1,   X,X,X,X]->[0,1,1,1];
[C,0,1,   X,X,X,X]->[0,0,0,0];
[C,0,1,   X,X,X,X]->[1,0,0,0];

END GLOBAL_TIMEOUT_CHIP
```

BWCOUNT.abl BLOCK COUNTER AND WORD COUNTER CONTROL

```
MODULE BLK_AND_WRD_COUNTER_CONTROL
FLAG '-R7'
TITLE 'ANDY ROMERO RD/EE CONTROLS 2-13-90'

BWCOUNTC DEVICE 'P20V8R';

PIN1,PIN23,PIN2,PIN3,PIN4,PIN5,PIN6,PIN7,PIN8,PIN9,PIN10,PIN14  PIN  1,23,2,3,4,5,6,7,8,9,10,14;
PIN15,PIN17,PIN18,PIN19,PIN20,PIN21,PIN22  PIN 15,17,18,19,20,21,22;

"Declaration of Active Levels"

CLK = PIN1;        CNT_RD = !PIN22;
BLKWRD = !PIN23;   INC_BC = !PIN21;
CSRC = !PIN2;      BCWRT = !PIN20;
FB_WR = !PIN3;     WCS0 = PIN19;
CLKB = PIN4;       WCS1 = PIN18;
EVENTEN = !PIN5;   INC_WC = !PIN17;
DONE = !PIN6;      WCDONE = !PIN15;
RESET = !PIN7;
FB_RD = !PIN8;
COVFLOW = !PIN9;
NOTRUNC = !PIN10;
PIPEHOLD =!PIN14;

"End Declaration of Active Levels"


"Standard Symbol Declarations"

X=.X.;
C=.C.;
P=.P.;
Z=.Z.;

"End Standard Symbol Declarations"

WCSELVEC = [WCS1, WCS0];
CLEAR_WC = [0,0];
COUNT_DOWN = [0,1];
LOAD_WC = [1,0];
COUNT_UP = [1,1];

EQUATIONS

CNT_RD = CSRC & FB_RD;

INC_BC = (
        (BLKWRD)
        #(CSRC & FB_WR)
       );

BCWRT = (
        (CSRC & FB_WR)
       #(BCWRT & INC_BC)
      );

WHEN (RESET # BLKWRD) THEN WCSELVEC = CLEAR_WC;
ELSE WHEN (CSRC & FB_WR) THEN WCSELVEC = LOAD_WC;
```

BWGATEC.abl STAGE 2 GATING CHIP

```
MODULE STAGE_2_GATING_CHIP
FLAG '-R1'
TITLE 'ANDY ROMERO RD/EE CONTROLS 2-13-90'

BWGATEC DEVICE 'P20V8R';

PIN2,PIN3,PIN4,PIN7,PIN8,PIN9  PIN  2,3,4,7,8,9;
PIN22,PIN21,PIN20,PIN19,PIN18,PIN17  PIN  22,21,20,19,18,17;

"Declaration of Active Levels"

CLKB = PIN2;        DMHIW = PIN22;
EVENTEN = !PIN3;    DMLOW = PIN21;
PIPEHOLD = !PIN4;   P2W = PIN20;
                    DMLOGAT = !PIN19;
                    DMHIGAT = !PIN18;
S1 = PIN7;          P2GAT = !PIN17;
S0 = PIN8;
DONE = !PIN9;

"End Declaration of Active Levels"


"Standard Symbol Declarations"

X=.X.;
C=.C.;
P=.P.;
Z=.Z.;

"End Standard Symbol Declarations"

EQUATIONS

P2W = P2GAT & CLKB & !PIPEHOLD;

DMHIW = DMHIGAT & CLKB & !PIPEHOLD;

DMLOW = DMLOGAT & CLKB & !PIPEHOLD;

P2GAT = (
      (EVENTEN & !CLKB & !DONE)
      #(P2GAT & CLKB)
      );

DMHIGAT = (
        (S1 & !CLKB)
        #(S1 & DMHIGAT)
        #(CLKB & DMHIGAT)
        );

DMLOGAT = (
        (S0 & !CLKB)
        #(S0 & DMLOGAT)
        #(CLKB & DMLOGAT)
        );

END STAGE_2_GATING_CHIP
```

```
FIFGAT = (
        (FWRTRDY & !CLKB)
       #(FIFGAT & FWRTRDY)
       #(FIFGAT & CLKB)
       );

FWRTRDY := (
        (DMHIGAT)
       #(FWRTRDY & PIPEHOLD0) "Save the state of FWRTRDY if Pipehold"
       );

HOLDPIP := (
        (AUXSEL & AUXHALF)
       #(FASTSEL & EVBHF)
       );

PIPEHOLD0 = (
        (((!AUXSEL & STOPEVBW) # HOLDPIP) & !CLKB)
       #(PIPEHOLD0 & ((!AUXSEL & STOPEVBW) # HOLDPIP))
       #(PIPEHOLD0 & CLKB)
       );

PIPEHOLD1 = (
        (((!AUXSEL & STOPEVBW) # HOLDPIP) & !CLKB)
       #(PIPEHOLD1 & ((!AUXSEL & STOPEVBW) # HOLDPIP))
       #(PIPEHOLD1 & CLKB)
       );

PIPEHOLD2 = (
        (((!AUXSEL & STOPEVBW) # HOLDPIP) & !CLKB)
       #(PIPEHOLD2 & ((!AUXSEL & STOPEVBW) # HOLDPIP))
       #(PIPEHOLD2 & CLKB)
       );

END OUTPUT_FIFO_WRITE_CONTROL
```

BWSTATE.abl PIPELINE CONTROL STATE MACHINE

```
MODULE PIPELINE_CONTROL_STATE_MACHINE

FLAG   '-r7'

TITLE  'Andy Romero  RD/EE Controls 2-13-90'


"Modification 7-24-90
"Modify this chip to include an output that will allow
"one event to be written to the Fastbus Event Buffer and then
"prevent further writes until the previous event has been read from the
"Event Buffer by FASTBUS. This circuit will only allow complete events to
"be writen to the event buffer. The following signals were added
"FBFEF,STOPEVBW.


BWSTATEM DEVICE        'P20V8R';

PIN1,PIN3,PIN4,PIN5,PIN6,PIN7,PIN8,PIN9  PIN  1,3,4,5,6,7,8,9;
PIN19,PIN18,PIN17,PIN16,PIN15  PIN  19,18,17,16,15;

"DECLARATION OF ACTIVE LEVELS"

CLK = PIN1;        S3 = PIN19;
NOTRUNC = !PIN3;   S2 = PIN18;
COVFLOW = !PIN4;   S1 = PIN17;
RESET = !PIN5;     S0 = PIN16;
DONE = !PIN6;      STOPEVBW = PIN15;
EVENTEN = !PIN7;
PIPEHOLD = !PIN8;
FBFEF = !PIN9;

"END DECLARATION OF ACTIVE LEVELS"

"STANDARD SYMBOL DECLARATIONS"

X=.X.;
C=.C.;
P=.P.;
Z=.Z.;

"END STANDARD SYMBOL DECLARATIONS"

@INCLUDE 'E:\\PROJECTS\\SSDSEQ\\PALS\\PIPSTATE.TBL'

"Declarations concerning STOPEVBW state machine
ALLOW = 0; "allow FASTBUS fifo writes
PREVENT = 1; "prevent  FASTBUS fifo writes

STATE_DIAGRAM STATEVEC

STATE WAIT:;

IF RESET THEN WAIT
ELSE IF DONE THEN LASTLO
ELSE IF (EVENTEN & !DONE) THEN DATLO
ELSE WAIT;
```

```
GOTO WAIT;

STATE TRAP3:;

GOTO WAIT;

STATE TRAP4:;

GOTO WAIT;

STATE TRAP5:;

GOTO WAIT;

STATE TRAP6:;

GOTO WAIT;

STATE_DIAGRAM STOPEVBW

STATE ALLOW:
IF ((STATEVEC == SET_CNT) & !FBFEF) THEN PREVENT
ELSE ALLOW;


STATE PREVENT:
IF ((STATEVEC == WAIT) & FBFEF) THEN ALLOW
ELSE PREVENT;

END PIPELINE_CONTROL_STATE_MACHINE
```

```
!CYCLE = (
        ((STATEVEC == WAIT) & !CLKB)
        #((STATEVEC == WAIT) & !CYCLE)
        #(CLKB & !CYCLE)
        );

WCNT_EN = (
        ((STATEVEC == FLUSH) & !CLKB)
        #((STATEVEC == FLUSH) & WCNT_EN)
        #(CLKB & WCNT_EN)
        );

END PIPELINE_STATE_DECODER
```

```
AUXSEL := 1; "KLUDGE FOR ABEL telling ABEL to make the cell registered.
AUXSEL.C = 1; "KLUDGE FOR ABEL telling ABEL to not clock with pin 1. (I really
        "only want to use the preset and clear functions and don't want
        "to clock the register at all.)


AUXSEL.RE = /CSR0 * /FB_WR * BAD7IN "Enable loading of Auxiliary output FIFO by
      + /RESET;                     " writing a 1 to CSR0<7>. Also enable at
                                    " reset time. /AUXSEL is active low.

AUXSEL.PR = /CSR0 * /FB_WR * BAD23IN;"Disable loading of FASTBUS output FIFO by
                                    " writing a 1 to CSR0<23>.
                                    " writing a 1 to CSR0<23>.

BAD7 = /AUXSEL.Q; " Status of Auxiliary Output FIFO loading is read
             " back on CSR0<7> via BAD(7)

BAD7.OE = /CSR0 * /FB_RD; " Enable active high status bit onto BAD(7)
                    " that indicates if the Auxiliary output FIFO is
                    " enabled/disabled as part of CSR0 read operation.

NOTRUNC := 1; "KLUDGE FOR ABEL telling ABEL to make the cell registered.
NOTRUNC.C = 1; "KLUDGE FOR ABEL telling ABEL to not clock with pin 1. (I really
        "only want to use the preset and clear functions and don't want
        "to clock the register at all.)


NOTRUNC.PR = /CSR0 * /FB_WR * BAD8   " Disable active low NOTRUNC which is
      + /RESET;                      " used to prevent truncating events at
                                     " 255 when it is not enabled.

NOTRUNC.RE = /CSR0 * /FB_WR * BAD24IN; " Enable active low NOTRUNC so that
                                     " events will NOT be truncated at 255.

BAD8      = NOTRUNC.Q;                    " Status of NOTRUNC will be read back BAD(8)
                          " of CSR0.

BAD8.OE = /CSR0 * /FB_RD; " Enable active high status bit onto BAD(8)
                    " that indicates if the Overflow Truncation is
                    " enabled/disabled as part of CSR0 read operation.

SYNSTAT.C = /TSYNERR; " Clock the SYNC status F/F with TTL level Sync Error

/SYNSTAT := 1; " When clocked by TSYNERR, assert /SYNSTAT

SYNSTAT.PR = SYNRESFF.Q * FB_RD "Clear ACTIVE LOW /SYNSTAT, hence not .RE
      + /RESET;


SYNRESFF := 1; "The Sync Error Reset F/F will have a 1 clocked in when
           "FASTBUS reads the CSR at C000_0012. This is in preparation
           "for clearing the F/F at the end of the read.

SYNRESFF.C = /C000_0012 * /FB_RD; "Clock the Sync Error Reset F/F when FASTBUS
                          "reads from CSR at C000_0012.

SYNRESFF.RE = SYNRESFF.Q * /C000_0012 * FB_RD "Clear the Sync Error Reset F/F
      + /RESET;                       "when FASTBUS finishes that read
                                      "that set the F/F originally.
```

ENCEN.abl ENCODER FIFO CIRCUIT CONTROL

```
MODULE              Enc_Fifo_Ckt_Pal

FLAG                '-r7'

TITLE               ' Andy Romero  RD/EE Controls'

ENCEN               DEVICE          'P20V8R';

PIN1,PIN2,PIN3,PIN4,PIN6,PIN8,PIN9,PIN10,PIN11  PIN  1,2,3,4,6,8,9,10,11;
PIN14,PIN15,PIN16,PIN17,PIN18,PIN19  PIN  14,15,16,17,18,19;
PIN20,PIN21,PIN22   PIN  20,21,22;

"STANDARD SYMBOL DECLARATIONS

X=.X.;
C=.C.;
P=.P.;


"DECLARATION OF ACTIVE LEVELS AND DESCRIPTION OF PIN FUNCTIONS

SQCLK = PIN1;          FW = PIN22;
SQCLKB = PIN2;         EBUSY = !PIN21;
DT_VAL = PIN3;         FERR = !PIN20;
ENCOTMO = !PIN4;       EFSM2 = PIN19;
DLYRST = !PIN6;        EFSM1 = PIN18;
PRELOAD = !PIN8;       EFSM0 = PIN17;
PIPEHOLD = !PIN9;      RDGAT = !PIN16;
RDEN = !PIN10;         FRD = PIN15;
FCLK = PIN11;
FF = !PIN14;

EFSTATE = [EFSM2,EFSM1,EFSM0];

S0 = [0,0,0];
S1 = [0,0,1];
S2 = [0,1,0];
S3 = [0,1,1];
S4 = [1,0,0];
S5 = [1,0,1];
S6 = [1,1,0];
S7 = [1,1,1];


EQUATIONS

FW = (
     (
       ( SQCLKB & EBUSY ) "Normal data write
      # ( SQCLKB & ENCOTMO & ( EFSTATE==S0 ) ) "Timeout EOE write
     )
     & (!FF & !FERR) "Fifo not full"
    )
  # (DLYRST); "Hold high on reset

EBUSY =  (!SQCLKB & DT_VAL) "Clk qualify EBUSY for glitch free gating of FW"
     # (EBUSY & DT_VAL) "Latch gating while DT_VAL active"
     # (EBUSY & ((EFSTATE==S1) # (EFSTATE==S3)))"Insure that EOE gets written"
```

```
STATE S7::

Goto S0;


TEST_VECTORS

"Test of Basic State Machine Operation"

([PIN1,    PIN3,  PIN4,  PIN6,   EFSTATE] -> [EFSTATE])
"([SQCLK,   DT_VAL, ENCOTMO, DLYRST,   EFSTATE] -> [EFSTATE])

[P, 0,1,1,  S0] -> [X];
[C, 0,1,1,  X] -> [S0];
[C, 1,1,1,  X] -> [S1];
[C, 1,0,1,  X] -> [S3];
[C, X,0,1,  X] -> [S2];
[C, X,X,X,  X] -> [S0];
[C, 0,1,1,  X] -> [S0];
[C, 1,1,1,  X] -> [S1];
[C, 0,1,1,  X] -> [S5];
[C, 0,0,1,  X] -> [S4];
[C, X,X,X,  X] -> [S0];
[C, 1,1,X,  X] -> [S0];
[C, 1,1,1,  X] -> [S1];
[C, 0,0,1,  X] -> [S0];

"Test of all transitions out of all states

"([PIN1,    PIN3,  PIN4,  PIN6,   EFSTATE] -> [EFSTATE])
"([SQCLK,   DT_VAL, ENCOTMO, DLYRST,   EFSTATE] -> [EFSTATE])

"S0
[P, 0,1,1,  S0] -> [X];
[C, 0,1,1,  X] -> [S0];
[C, X,X,0,  X] -> [S0];
[C, 1,1,1,  X] -> [S1];

"S1
[P, 1,1,1,  S1] -> [X];
[C, 1,1,1,  X] -> [S1];
[P, 1,1,1,  S1] -> [X];
[C, 1,0,1,  X] -> [S3];
[P, 1,1,1,  S1] -> [X];
[C, X,X,0,  X] -> [S0];
[P, 1,1,1,  S1] -> [X];
[C, 0,0,1,  X] -> [S0];
[P, 1,1,1,  S1] -> [X];
[C, 0,1,1,  X] -> [S5];

"S2
[P, 0,1,1,  S2] -> [X];
[C, X,X,X,  X] -> [S0];

"S3
[P, 1,1,1,  S3] -> [X];
[C, 1,1,1,  X] -> [S3];
[P, 1,1,1,  S3] -> [X];
[C, X,X,0,  X] -> [S2];
[P, 1,1,1,  S3] -> [X];
[C, 0,1,1,  X] -> [S2];
```

```
"this because there is no maximum limit for the width of the active low Fifo
"write signal

"IMPORTANT NOTE xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

"When designing the chips that generate the following signals, insure that
"they are only allowed to change state when FCLK or SQCLK are low.
"ENCOTMO, FF (determined by fifo looks ok), PRELOAD, PIPEHOLD

"IMPORTANT NOTE xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

""Redundant (glitch prevent) terms were added to the equations for EBUSY and
"and RDGAT; these terms will be eliminated by the ABEL Reduce program if any
"reduction level greater than -r1 is used. This would not be a problem except
"that the state machine (EFSM <2:0>) requires extensive reduction (-r7
"is highly recomended). Here is how to get around this problem.

"    1)Compile this file, ENCFIFO.ABL, with -r7 reduction
"    2)Copy ENCFIFO.ABL to TEMPFILE.ABL
"
" DO NOT DELETE ENCFIFO.ABL IT IS THE MASTER SOURCE DOCUMENT !!!!
"
"    3)Open TEMPFILE.ABL, change the flag statement to activate -r1
"      reduction. Delete the state machine description from TEMPFILE.ABL.
"      Paste the reduced equations from ENCFIFO.DOC for the state terms
"      (EFSM <2:0>) into the equations section of TEMPFILE.ABL.
"
"    4)Compile TEMPFILE.ABL to obtain a glitch free JED file
"
"    5)Delete TEMPFILE.*
"

"INPUTS

"SQCLK = PIN1;

"SQCLK is a clock signal derived from the Sequencer CLK3 signal; SQCLK is used
"to clock a small state machine within this pal.

"SQCLKB = PIN2;

"SQCLKB is the same signal as SQCLK; it provides a waveform of suitable duty
"cycle for writing the Encoder Fifos.

"DT_VAL = PIN3;

"(ACTIVE HIGH) DT_VAL is the Data Valid signal from the Delay Encoder
"Module.

"ENCOTMO = !PIN4;

"(ACTIVE LOW) The Sequencer module sends the Delay encoder module an Address
"Valid signal the Delay Encoder module responds, if it has data to send
"to the Sequencer module, by activating (and holding while it is sending data)
"the Data Valid signal. If the Delay Encoder has no data to send it does not
"issue a Data Valid. 300ns (8 SQCLK cycles) after Address Valid is sent, the
"FACU activates the ENCOTMO signal. If this pal has not received a Data Valid
"before ENCOTMO becomes active an EOE word is written into the Encoder Fifo

"DLYRST = !PIN6;
```

```
"read signal.

"FRD = PIN15;

"(ACTIVE HIGH) Encoder Fifo read
"signal


END Enc_Fifo_Ckt_Pal
```

PALFIF2.abl FIFO CONTROL FOR AUXILIARY INTERFACE

```
MODULE _PALFIF2 flag '-r3'

TITLE
'FIFO CONTROL FOR AUXILIARY INTERFACE TO TAXI '

    PALFIF2     DEVICE      'P16V8R'; "Use a 16R6B for 16 MHZ. 386

" CONSTANTS:

    ON      =       1;
    OFF     =       0;
    H       =       1;
    L       =       0;
    X       =       .X.;    " ABEL  don't care symbol
    C       =       .C.;    " ABEL clocking input symbol

" Pin names:
                                "INPUT PINS
    ACK     pin     2;          "ACKNOWLEDGE FROM TAXI
    CLOCK           pin     3;      "CLOCK
    AUXOR           pin     4;      "AUXILIARY output READY
    TXRDY           pin     5;      "TXRDY

    SOENA           pin     6;      "SHIFT OUT ENABLE
    STR     pin     7;          "STROBE FROM SEQUENCER
    NU8     pin     8;          "NOT USED 8
    NU9     pin     9;          "NOT USED 9

    CLK     pin     1;          "Clock
    NU11    pin     11;         "NOT USED 11
    TRIG    pin     12;         "TRIGGER ACK ONESHOT
    NU19    pin     19;         "NOT USED 19

                                "OUTPUT PINS
    STROBE          pin     18;     "STROBE TO TAXI
    NU17    pin     17;         "NOT USED 17
    NU16    pin     16;         "NOT USED 16
    TXQ     pin     15;         "OUTPUT READY STATE Q
    AUXRNOT         pin     14;     "READ OF FIFO
    NU13    pin     13;         "NOT USED 13

equations

STROBE = ((STR & !CLOCK) # (!CLOCK & !AUXRNOT & SOENA)
    # (STROBE & !CLOCK));
TXQ = ((CLOCK & TXQ) # (TXRDY & CLOCK));
AUXRNOT =!((TXRDY & SOENA & TXQ) # (!AUXRNOT & TXQ));
TRIG = (TXRDY & SOENA & TXQ);

END _PALFIF2
```

RDEN1.abl SMALL END-OF-EVENT DECODER

```
MODULE SMALL_EOE_DECODER
FLAG '-R7'
TITLE 'ANDY ROMERO RD/EE CONTROLS 2-6-90'

RDEN1 DEVICE 'P18N8';

EVENTEN,EOE1,EOE2,EOE3,EOE4,EOE5,EOE6,EOE7,EOE8 PIN  1,2,3,4,5,6,7,8,9;
READEN1,READEN2,READEN3  PIN  19,18,17;
READEN4,READEN5,READEN6  PIN  16,15,14;
READEN7,READEN8          PIN  13,12;

"Standard Symbol Declarations"

X=.X.;
C=.C.;
P=.P.;

"End Standard Symbol Declarations"

VECTOR = [EVENTEN,EOE1,EOE2,EOE3,EOE4,EOE5,EOE6,EOE7,EOE8];

EQUATIONS

!READEN1 = (VECTOR==[0,1,X,X,X,X,X,X,X]);
!READEN2 = (VECTOR==[0,0,1,X,X,X,X,X,X]);
!READEN3 = (VECTOR==[0,0,0,1,X,X,X,X,X]);
!READEN4 = (VECTOR==[0,0,0,0,1,X,X,X,X]);
!READEN5 = (VECTOR==[0,0,0,0,0,1,X,X,X]);
!READEN6 = (VECTOR==[0,0,0,0,0,0,1,X,X]);
!READEN7 = (VECTOR==[0,0,0,0,0,0,0,1,X]);
!READEN8 = (VECTOR==[0,0,0,0,0,0,0,0,1]);

TEST_VECTORS

([VECTOR] ->
[READEN1,READEN2,READEN3,READEN4,READEN5,READEN6,READEN7,READEN8])

[[1,X,X,X,X,X,X,X,X]] -> [1,1,1,1,1,1,1,1];
[[0,1,X,X,X,X,X,X,X]] -> [0,1,1,1,1,1,1,1];
[[0,0,1,X,X,X,X,X,X]] -> [1,0,1,1,1,1,1,1];
[[0,0,0,1,X,X,X,X,X]] -> [1,1,0,1,1,1,1,1];
[[0,0,0,0,1,X,X,X,X]] -> [1,1,1,0,1,1,1,1];
[[0,0,0,0,0,1,X,X,X]] -> [1,1,1,1,0,1,1,1];
[[0,0,0,0,0,0,1,X,X]] -> [1,1,1,1,1,0,1,1];
[[0,0,0,0,0,0,0,1,X]] -> [1,1,1,1,1,1,0,1];
[[0,0,0,0,0,0,0,0,1]] -> [1,1,1,1,1,1,1,0];


END SMALL_EOE_DECODER
```

SAMTX.abl TRANSMIT LOGIC FOR AUXILIARY INTERFACE

```
MODULE SAMTX_EMULATOR flag '-r3'

TITLE
'SEQUENCER AUXILIARY SAMTX PAL '

    SAMTX       DEVICE      'P16V8R'; "Use a SAM CHIP IN REAL BOARD

" CONSTANTS:

    ON      =       1;
    OFF     =       0;
    H       =       1;
    L       =       0;
    X       =       .X.;    " ABEL  don't care symbol
    C       =       .C.;    " ABEL clocking input symbol

" State definitions :

    ACTIVE              =       ^b000;  "Check for Stopreq and TXRDY
    NU1         =       ^b001;  "NOT USED
    NU2         =       ^b010;  "NOT USED
    NU3         =       ^b011;  "NOT USED
    INIT0       =       ^b100;  "Setup INIT and clear error status
    INIT1       =       ^b101;  "Send INIT and load loop count 20us.
                                "Waiting for RXINIT or timeout
    INQ0        =       ^b110;  "CLR status and setup INQ
    INQ1        =       ^b111;  "Load wait timer and send INQ
                                "Waiting for RXRDY,/RXNRDY or timeout


" Pin names:
                                "INPUT PINS
    TXRDY       pin     2;      "TRANSMITER READY data in fifo
    RXRDY       pin     4;      "READY from reciever
    RXNRDY      pin     5;      "NOT READY from reciever
    AUXERR      pin     7;      "ERROR on link

    STOPREQ     pin     6;      "STOP request form RX
    RXINIT      pin     3;      "Reciever has been initialized
    TMO   pin   8;      "20 microsec Timout
    NRESET      pin     9;      "RESET onmain board

    CLK2  pin   1;      "Clock
    OE    pin   11;     "output enable

                                "OUTPUT PINS
    STRout      pin     15;     "Control strobe
    SOENA       pin     12;     "Shift out and strobe enable
    SCLR  pin   17;     "Status register clear
    LINKRDY     pin     19;     "ready for data transfer
    SEQ   pin   18;     "SEQUENCE state control
    S0    pin   16;     "STATE code bit 0
    C1    pin   14;     "Control code bit 1
    C0    pin   13;     "Control code bit 0

"MAINSTATE definition.

    MAINSTATE = [C1, C0, S0];
```

```
        state NU1:                      "unused state - should not occur
         SEQ := OFF;
         SCLR := ON;
         STRout := OFF;
         SOENA := OFF;
         LINKRDY := OFF;
         if !NRESET then INIT0          "reset to INITstate
         else NU2;                      "if entered go to INITstate

        state NU2:                      "unused state - should not occur
         SEQ := OFF;
         SCLR := ON;
         STRout := OFF;
         SOENA := OFF;
         LINKRDY := OFF;
         if !NRESET then INIT0          "reset to INITstate
         else NU3;                      "if entered go to INITstate

        state NU3:                      "unused state - should not occur
         SEQ := OFF;
         SCLR := ON;
         STRout := OFF;
         SOENA := OFF;
         LINKRDY := OFF;
         if !NRESET then INIT0          "reset to INITstate
         else INIT0;                    "if entered go to INITstate

end SAMTX_EMULATOR;
```

```
" CHANGE HISTORY:
" 4-30-90    : Pin 14 was changed from EMPTY to EVBEF.
" 4-30-90    : The device type was changed from 20L8 to 20V8.
" 6-05-90    : FB_RD & !EVBEF was added to FIFO equation.
" 6-20-90    : EOB was modified to react to a change on the EVBEF signal
"             : only when FB_RD is inactive in order to prevent EOB being
"             : issued DURING the read of the last word in the FIFO.
" 6-20-90    : EOB before change:      EOB = CON & !CSR & EVBEF
"                                           # CON & CSR & INC & NTV  ;
" 6-20-90    : The p-term (FIFO * FB_RD) was added to FIFO to latch it.

END SEQ_NTA
```

```
            "Block transfer DS dn
        # CON & INC_LD & !INH & IRD & !IMS2 & IMS1 & !IMS0
            & IDS & !DIDS2 & !NTACLK   "NTA read cycle
        # CON & INC_LD & !INH & !IMS2 & !IMS1 & !IMS0
            & IDS & !DIDS2 & !NTACLK;  "Single R/W cycle

"INCrement or LoaD the NTA

 INC_LD = !(!(CON & !INH & IMS0 & IDS & DIDS2) & !INC_LD "Set if blk
     # !IRD & !IMS2 & IMS1 & !IMS0 & IDS & !DIDS2 "Reset on NTA write
     # !IMS2 & !IMS0 & NTACLK  "Reset on NTA or Single & after NTACLK
     # !CON ) ;        "Reset when disconnected

"Output AK

 OAK = CON & !IAK & !IWT                    "Send AK if not broadcast address
     " Removed !BCADD from the first p-term on 4-30-90. The Sequencer
     " doesn't support broadcast
     # OAK & CON                    "Latch until CON goes away
     # OAK & IWT                    "Hold if WaiT is asserted
     # OAK & DIDS2  ;                    "Stretch until DK is off

OWT = DIAS & CON & !IMS2 & !IMS1 & !IMS0 & !OAK & EVBEF
    + OWT & DIAS & EVBEF; "Assert WAIT when the FIFO is empty at address
                          "strobe time and data space is being addressed.
                          "Latch until FIFO goes not empty or master
                          "disconnects by removing AS.

" CHANGE HISTORY:
" 4-30-90     : OWT added to originally unused pin 15 and set to 0.
" 4-30-90     : BCADD was changed to EVBEF.  BCADD is for broadcast
"              which the Sequencer doesn't support.  EVBEF is for WAIT
" 4-30-90     : Device type changed from 20L8 to 20V8
" 5-2-90      : Polarity of IRD changed to active low.
" 7-18-90     : OWT is implemented to assert WAIT at AS time.
" 7-18-90     : The device was changed from 20V8 to 22V10. Needed feedbacks.
" 7-23-90     : Change polarity of input EVBEF to !EVBEF.
END SEQ_PAL1
```

```
"Output Slave Status bit 0

 OSS0 = CON & BSY  & !IMS2 & !IMS1 & !IMS0 & DIDS1 & !DIDS2 & !CSR "BuSY & single xfer
    # CON & NTV & !IMS2 & IMS1 & !IMS0 & DIDS1 & !DIDS2 "NTV & 2nd Address
    # OSS0 & IDS & DIDS2 & CON                    "Latch while DS=DK=1
   # OSS0 & !IMS2 & IMS0 & !IDS & !DIDS2 & CON ; "Latch if DS=DK=0 & MS=1or3

"Output Slave Status bit 1
 OSS1 = CON & IMS2 & !CSR  "any MS=4-7 on DS up or down
    # CON & !IMS2 & NTV & DIDS1 & !DIDS2
    "any Not Valid address if not BuSY on DS up
    # CON & !IMS2 & IMS0 & NTV & !EOB & !DIDS1 & DIDS2
    "any Not Valid address, not BuSY, not End Of Block, & MS=1or3 on DS down
    # CON & !IMS2 & IMS0 & EOB & DIDS1 & !DIDS2
    "End Of Block if not BuSY on DS up
    # CON & !IMS2 & IMS0 & EOB & !DIDS1 & DIDS2
    "End Of Block if not BuSY on DS down
    # OSS1 & IDS & DIDS2 & CON                    "Latch while DS=DK=1
    # OSS1 & IMS0 & !IDS & !DIDS2 & CON; "Latch if DS=DK=0 & MS=1,3,5,or 7

"Slave Status bit 2
 OSS2 = CON & IMS2   "any MS=4-7 on DS up or down
    # CON & !IMS2 & NTV & !EOB & DIDS1 & !DIDS2
    "any Not Valid address if not BuSY and not End Of Block on DS up
    # CON & !IMS2 & IMS0 & NTV & !EOB & !DIDS1 & DIDS2
    "any Not Valid address, not BuSY, not End Of Block, & MS=1or3 on DS down
    # OSS2 & IDS & DIDS2 & CON                    "Latch while DS=DK=1
    # OSS2 & IMS0 & !IDS & !DIDS2 & CON  ; "Latch if DS=DK=0 & MS=1,3,5,or 7

"Output Data acKnowlege generates DK

 ODK = !IWT & CON & OAK & DIDS2  "set if DS (delayed) and attached and not WaiT
    # IMS1 & IMS0 & CON & OAK & DIDS2
    "set if DS (delayed) and attached and MS=3 (pipeline) even if WaiT
   # ODK & CON & OAK & DIDS2  "transition hold while DS (delayed)
    # IWT & !IMS1 & ODK   "hold if WaiT and not MS1
    # IWT & !IMS0 & ODK  ;          "hold if WaiT and not MS0
    "i.e. hold if not MS=3 (pipeline) AND WaiT, release it otherwise

"FastBus ReaD

 FB_RD = CON & !INH & IRD & !IMS2 & !IMS1 & !IMS0 & DIDS1 & !DIDS2
          "set on DS up, MS=0 random data read
    # CON & !INH & IRD & !IMS2 & IMS0 & DIDS1 & !DIDS2
          "set on DS up, MS=1 block read
          "set on DS up, MS=3 pipeline read
    # CON & !INH & IRD & !IMS2 & IMS0 & !DIDS1 & DIDS2
          "set on DS dn, MS=1 block read
          "set on DS dn, MS=3 pipeline read
    # CON & IRD & !IMS2 & !IMS1 & !IMS0 &IDS & DIDS2 & FB_RD
          "latch while MS=0 read and DS,DK up
          "latch while MS=1,3 read
    # CON & IDS & DIDS2 & FB_RD                   "latch while DS,DK up
    # CON & !IDS & !DIDS2 & FB_RD  ;     "latch while DS,DK down
                                  "i.e. latch until new MS or WR cycle

"Output INHibit data transfers

 OINH = INH
    # CON & !CSR ;          "This stops NTA Incrementing for the FIFO in Data Space
```

TRIGCNT.abl TRIGGER COUNTER

```
MODULE TRIGGER_COUNTER
FLAG '-R7'
TITLE 'ANDY ROMERO RD/EE CONTROLS 6-14-90'


TRIGCNT DEVICE 'P22V10';

PIN1,PIN3,PIN4,PIN5   PIN  1,3,4,5;

PIN23,PIN22,PIN21,PIN20,PIN19  PIN  23,22,21,20,19;
PIN18,PIN17,PIN16,PIN15,PIN14  PIN  18,17,16,15,14;

"Declaration of active levels"

CLK25MHZ    = PIN1;
TRINC       = !PIN3;
DEC         = !PIN4;
RESET       = !PIN5;

NOTEMP      = !PIN23;
SYNCINC     = PIN22;
HF          = !PIN21;
ERROR       = !PIN20;
TRCQ5       = PIN19;
TRCQ4       = PIN18;
TRCQ3       = PIN17;
TRCQ2       = PIN16;
TRCQ1       = PIN15;
TRCQ0       = PIN14;

"Standard Symbol Declarations"

X=.X.;
C=.C.;
P=.P.;

TRIG_COUNT = [TRCQ5,TRCQ4,TRCQ3,TRCQ2,TRCQ1,TRCQ0];
WAIT_FOR_TRINC = 1;
SYNCINC_ACTIVE = 0;
OVERFLOW = [1,1,1,1,1,1];

EQUATIONS

WHEN RESET THEN TRIG_COUNT := [0,0,0,0,0,0];
ELSE WHEN (TRIG_COUNT == OVERFLOW) THEN TRIG_COUNT := OVERFLOW;
ELSE WHEN ((SYNCINC == SYNCINC_ACTIVE) & DEC) THEN TRIG_COUNT := TRIG_COUNT;
ELSE WHEN ((SYNCINC == SYNCINC_ACTIVE) & !DEC) THEN TRIG_COUNT := (TRIG_COUNT + 1);
ELSE WHEN ((SYNCINC == WAIT_FOR_TRINC) & DEC) THEN TRIG_COUNT := (TRIG_COUNT - 1);
ELSE TRIG_COUNT := TRIG_COUNT;


HF := ((TRIG_COUNT >= 31) & !RESET);

ERROR := ((TRIG_COUNT == 63) & !RESET);

NOTEMP = (TRIG_COUNT != 0);
```

```
[C,  0,1,1]->[0,0,1,1,  19];
[C,  X,1,1]->[0,1,1,1,  20];
[C,  1,0,1]->[0,1,1,1,  19];
[C,  1,0,1]->[0,1,1,1,  18];
[C,  1,0,1]->[0,1,1,1,  17];
[C,  1,0,1]->[0,1,1,1,  16];

[C,  0,1,1]->[0,0,1,1,  16];
[C,  X,1,1]->[0,1,1,1,  17];
[C,  0,1,1]->[0,0,1,1,  17];
[C,  X,1,1]->[0,1,1,1,  18];
[C,  0,1,1]->[0,0,1,1,  18];
[C,  X,1,1]->[0,1,1,1,  19];
[C,  0,1,1]->[0,0,1,1,  19];
[C,  X,1,1]->[0,1,1,1,  20];

"[CLK      /TRNC /DEC /RST ]-> [/ NTMP  SINC  /HF   ERR   TR_CNT]

[C,  0,1,1]->[0,0,1,1,  20];
[C,  X,1,1]->[0,1,1,1,  21];
[C,  0,1,1]->[0,0,1,1,  21];
[C,  X,1,1]->[0,1,1,1,  22];
[C,  0,1,1]->[0,0,1,1,  22];
[C,  X,1,1]->[0,1,1,1,  23];
[C,  0,1,1]->[0,0,1,1,  23];
[C,  X,1,1]->[0,1,1,1,  24];
[C,  0,1,1]->[0,0,1,1,  24];
[C,  X,1,1]->[0,1,1,1,  25];
[C,  0,1,1]->[0,0,1,1,  25];
[C,  X,1,1]->[0,1,1,1,  26];
[C,  0,1,1]->[0,0,1,1,  26];
[C,  X,1,1]->[0,1,1,1,  27];
[C,  0,1,1]->[0,0,1,1,  27];
[C,  X,1,1]->[0,1,1,1,  28];
[C,  0,1,1]->[0,0,1,1,  28];

"[CLK      /TRNC /DEC /RST ]-> [/ NTMP  SINC  /HF   ERR   TR_CNT]

[C,  X,1,1]->[0,1,1,1,  29];
[C,  0,1,1]->[0,0,1,1,  29];
[C,  X,1,1]->[0,1,1,1,  30];
[C,  0,1,1]->[0,0,1,1,  30];
[C,  X,1,1]->[0,1,1,1,  31];
[C,  0,1,1]->[0,0,0,1,  31];
[C,  X,1,1]->[0,1,0,1,  32];
[C,  0,1,1]->[0,0,0,1,  32];
[C,  X,1,1]->[0,1,0,1,  33];
[C,  0,1,1]->[0,0,0,1,  33];
[C,  X,1,1]->[0,1,0,1,  34];
[C,  0,1,1]->[0,0,0,1,  34];
[C,  X,1,1]->[0,1,0,1,  35];
[C,  0,1,1]->[0,0,0,1,  35];
[C,  X,1,1]->[0,1,0,1,  36];
[C,  0,1,1]->[0,0,0,1,  36];
[C,  X,1,1]->[0,1,0,1,  37];
[C,  0,1,1]->[0,0,0,1,  37];
[C,  X,1,1]->[0,1,0,1,  38];
[C,  0,1,1]->[0,0,0,1,  38];
[C,  X,1,1]->[0,1,0,1,  39];
[C,  0,1,1]->[0,0,0,1,  39];
[C,  X,1,1]->[0,1,0,1,  40];
```

```
[C,  0,1,1]->[0,0,0,1,  49];
[C,  X,1,1]->[0,1,0,1,  50];
[C,  0,1,1]->[0,0,0,1,  50];
[C,  X,1,1]->[0,1,0,1,  51];
[C,  0,1,1]->[0,0,0,1,  51];
[C,  X,1,1]->[0,1,0,1,  52];
[C,  0,1,1]->[0,0,0,1,  52];
[C,  X,1,1]->[0,1,0,1,  53];
[C,  0,1,1]->[0,0,0,1,  53];
[C,  X,1,1]->[0,1,0,1,  54];
[C,  0,1,1]->[0,0,0,1,  54];
[C,  X,1,1]->[0,1,0,1,  55];
[C,  0,1,1]->[0,0,0,1,  55];
[C,  X,1,1]->[0,1,0,1,  56];
[C,  0,1,1]->[0,0,0,1,  56];
[C,  X,1,1]->[0,1,0,1,  57];
[C,  0,1,1]->[0,0,0,1,  57];
[C,  X,1,1]->[0,1,0,1,  58];
[C,  0,1,1]->[0,0,0,1,  58];
[C,  X,1,1]->[0,1,0,1,  59];
[C,  0,1,1]->[0,0,0,1,  59];
[C,  X,1,1]->[0,1,0,1,  60];
[C,  0,1,1]->[0,0,0,1,  60];
[C,  X,1,1]->[0,1,0,1,  61];
[C,  0,1,1]->[0,0,0,1,  61];
[C,  X,1,1]->[0,1,0,1,  62];
[C,  0,1,1]->[0,0,0,1,  62];
[C,  X,1,1]->[0,1,0,1,  63];
[C,  0,1,1]->[0,0,0,0,  63];
[C,  0,X,1]->[0,1,0,0,  63];
[C,  1,X,1]->[0,1,0,0,  63];
[C,  X,X,0]->[1,1,1,1,  0];

"[CLK       /TRNC /DEC /RST ]-> [/ NTMP  SINC  /HF   ERR   TR_CNT]

END TRIGGER_COUNTER
```

APPENDIX C

PARTS LIST

BROOKTREE BT501KC	$15.00	4	$60.00
CYPRESS CY7C190-15PC	$9.25	2	$18.50
CON/WIN S15R8 25 MHZ OSC	$7.00	1	$7.00
1N4005	$0.04	1	$0.04
1N914	$0.03	1	$0.03
2N5770	$0.17	2	$0.34
51 OHM 1/8W RES	$0.18	2	$0.36
100 OHM 1/8W RES	$0.16	18	$2.88
130 OHM RES	$0.06	1	$0.06
220 OHM RES	$0.07	12	$0.84
330 OHM RES	$0.07	7	$0.49
1K RES	$0.06	11	$0.66
2K RES	$0.06	1	$0.06
2.2K RES	$0.06	29	$1.74
4.7K RES	$0.07	6	$0.42
10K RES	$0.06	3	$0.18
51K RES	$0.07	1	$0.07
DALE CSC08A-03-101G sipp	$0.30	51	$15.30
DALE CSC10A-01-101G sipp	$0.35	10	$3.50
DALE CSC06A-01-471G sipp	$0.22	2	$0.44
DALE CSC10A-01-102G sipp	$0.22	4	$0.88
33 uf CAP	$0.45	1	$0.45
47 uf CAP	$0.98	5	$4.90
68 pf CAP	$0.17	3	$0.51
SPRAGUE 923CZ5U104M050B 0.1uF CAP	$0.11	274	$30.69
GRAYHILL 76-RSB06S DIP SWITCH	$3.50	1	$3.50
3M 34PIN HEADER 3431-1303	$1.60	1	$1.60
5A PICOFUSE	$0.48	7	$3.36
MIL MAX 0665-0-15-15-30-27-10-2	$0.22	14	$3.08
Rgt Angle Coax LEMO conn. EPL. 00.250.NTN	$4.46	4	$17.84
LEDTRONICS RED PC120TR4 HI-EFFIC.	$0.45	10	$4.50
CK SWITCH EP12D1ABE	$5.36	1	$5.36
GEN. SEMI. ICTE-5 TRANSORB	$0.90	3	$2.70
AUXILIARY CONN			$0.00
FASTBUS CONN.			$0.00
FRONT PANEL			$0.00
HARDWARE			$0.00
ROB. NUGNT. ICT-243-S-TG 24 PIN IC SKT.	$0.88	26	$22.88
ROB. NUGNT. ICT-203-S-TG 20 PIN IC SKT.	$0.88	6	$5.28
ROB. NUGNT. ICA-143-SCO-TG30 OSC. SKT	$0.45	1	$0.45
BURNDY QILE 28P-410T 28 PIN PLCC SKT	$1.84	4	$7.36